Universidade de Aveiro Departamento de Comunicação e Arte
2010

Bruno Carlos Ferreira Dias Alves

CONSTRUÇÃO DE MÓDULOS PARA O DESENVOLVIMENTO ÁGIL DE SITES WEB NA DREAMLAB

Universidade de Aveiro Departamento de Comunicação e Arte
2010

Bruno Carlos Ferreira Dias Alves

Construção de módulos para o desenvolvimento ágil de sites Web na Dreamlab

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Comunicação Multimédia, realizada sob a orientação científica da Doutora Ana Margarida Pisco Almeida, Professora Auxiliar do Departamento de Comunicação e Arte da Universidade de Aveiro

Dedico este trabalho aos meus pais e irmã.

o júri

presidente

Prof. Doutor Pedro Alexandre Ferreira Santos Almeida

professor auxiliar no Departamento de Comunicação e Arte da Universidade de Aveiro

Prof. Doutor Joaquim Manuel Henriques de Sousa Pinto

professor auxiliar no Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

Prof. Doutora Ana Margarida Pisco Almeida

professor auxiliar no Departamento de Comunicação e Arte da Universidade de Aveiro

agradecimentos

Agradeço à minha família e amigos toda a força e apoio durante esta fase. Agradeço também à minha orientadora Professora Margarida Almeida pelo apoio e disponibilidade ao longo do desenvolvimento deste trabalho e cujo acompanhamento constante me manteve no caminho certo. Agradeço ainda ao Professor Francislê Neri de Souza pela ajuda e disponibilidade e não podia deixar de agradecer a toda a equipa da Dreamlab, principalmente à Clara Moreira e Maria João, pelo acompanhamento e toda a ajuda prestada já em contexto empresarial.

palavras-chave serviços, módulos, reutilização, sites web

resumo

A presente investigação tem como principal objectivo permitir agilizar o processo de criação de sites Web, no contexto específico da empresa Dreamlab, através da conceptualização e desenvolvimento de serviços chave de forma modular. A problemática do desenvolvimento ágil revelou-se particularmente relevante no contexto desta empresa uma vez que se procurava uma forma de conseguir diminuir os recursos necessários para a criação de conteúdos, sem haver diminuição da qualidade nos serviços prestados. Esta investigação focou-se, pois, na temática da agilidade e reutilização de conteúdos, em contexto online. Com vista ao desenvolvimento do trabalho realizado foi necessário compreender como funciona a modularidade, assim como todos os processos necessários que antecedem a criação de um módulo. Com os resultados obtidos, descritos nesta dissertação, espera-se dar um contributo válido no campo da agilidade de desenvolvimento para a Web, mas, mais especificamente, permitir à Dreamlab uma mais rápida capacidade de resposta aos pedidos que lhe são efectuados, através do soluções prontas a aplicar.

keywords services, module, reuse, web sites

abstract This investigation's main goal is to make web site's creation process more agile, specifically in Dreamlab, through the design and development of certain key services, in a modular way. Agile development is a relevant theme is this firm because of its search for a way to lower the resources needed to develop web content, without sacrificing the provided services' overall quality. This subject addresses issues like agility and content reuse, all in an online context. To develop this kind of work, it was necessary to first understand how modularity works, just like all the proceeding processes needed to develop a viable working module. With the end results detailed in this thesis, a viable contribute in this field is expected, and more specifically, to allow Dreamlab to address clients requests more rapidly, by way of ready to deploy solutions.

Índice

I. Introdução e enquadramento geral do problema de investigação 1
1. Apresentação 1
2. Questão de investigação 2
3. Finalidade e objectivos 3
II. Enquadramento teórico 4
1. Introdução 4
2. Programação para a Web 4
a. breve evolução histórica 4
b. definição de conceitos 6
3. Agilidade 8
a. revolução ágil 8
b. visão e inovação 9
c. modelo de desenvolvimento ágil 11
4. Arquitectura Orientada a Serviços 12
a. o que é 12
i. princípios gerais 13
ii. efeito de caixa negra 14
b. como funciona 15
c. condições ideais 15
d. soa governance 16
e. web services 16
5. Desenvolvimento Orientado a Aspectos 17
a. o que é 17
b. como funciona 18
c. separação de assuntos 19
6. Modularidade 19
a. como surgiu 19
b. objectivos gerais 20
c. como funciona 20
i. hierarquia 21
ii. abstracção 22
iii. generalidade 22

d. frameworks 22
III. Projecto em contexto empresarial 24
1. Procedimento metodológico 24
2. Etapas do estudo 26
3. Modelo de análise 27
4. Recolha de dados 28
5. Análise preliminar das práticas da empresa 30
6. Proposta e desenvolvimento dos módulos 32
a. Módulos escolhidos 32
b. Protótipo piloto: newsletter 34
i. Requisitos funcionais 34
ii. Desenvolvimento 34
c. Protótipo final: plataforma de partilha 42
i. Requisitos funcionais 42
ii. Desenvolvimento 42
d. Validação 52
e. Análise crítica dos protótipos desenvolvidos 53
f. Proposta de modelo de representação algorítmica 56
IV. Conclusões 59
1. Limitações do estudo 59
2. Perspectivas de trabalho futuro 60
3. Conclusões 61
Referências Bibliográficas 62
Anexos 67
1. Código fonte 67
a. Protótipo de *newsletter* 67
i. Newsletter.html 67
ii. Listar.php 68
iii. Inserir.php 68
iv. Enviar.php 69
v. Cancelar1.php 71
vi. Cancelar2.php 72
b. Protótipo de plataforma de partilha de ficheiros 72
i. Main.php 72

ii. Upload.php....73
iii. Listar.php....74
iv. Apagar.php....75
v. Login.php....76
vi. Logout.php....77
2. Guiões das entrevistas....78
a. Entrevista elementos da equipa de programação....78
b. Entrevista ao gestor de projectos....82

Índice de Ilustrações

Ilustração 1 - Framework de desenvolvimento ágil....11
Ilustração 2 - Protótipo de newsletter....35
Ilustração 3 - Protótipo da plataforma de partilha....43
Ilustração 4 - Representação visual do protótipo de *newsletter*....57
Ilustração 5 - Representação visual do protótipo da plataforma de partilha de ficheiros....58

I. Introdução e enquadramento geral do problema de investigação

1. Apresentação

Em contexto empresarial, é actualmente imperativo procurar por soluções optimizadas que permitam uma maior capacidade de resposta aos pedidos dos clientes. O número de pedidos efectuados às empresas multimédia tem aumentado não só em volume, mas também em complexidade e estas têm que garantir uma capacidade de resposta igual e ainda manter o nível de qualidade no serviço prestado. Há ainda a questão da concorrência, cuja tendência aponta para um constante crescimento nesta área, e que não pode ser ignorada no actual cenário global. Para garantir uma constante relevância no mercado e não perder terreno para eventuais concorrentes, cabe a cada empresa incentivar a procura por soluções que lhe permitam poupar tempo, recursos humanos, tecnológicos e dinheiro. Estas soluções implicam invariavelmente uma nova abordagem ao processo de desenvolvimento de serviços, não só tecnologicamente, mas também ao nível das mentalidades dos envolvidos. Relativamente ao processo de desenvolvimento, a prática mais frequente, e temática central do estudo realizado, passa pela divisão do todo em partes individuais, ou seja, encarar o problema total como um conjunto de sub-problemas mais pequenos e objectivos. Deste modo um produto multimédia completo passa a ser compreendido como a soma das suas partes, diferentes serviços mais ou menos independentes que, quando aplicados em conjunto, criam um produto final mais eficiente e mais facilmente reutilizado. Com esta divisão, cada serviço passa a ser compreendido como um módulo independente, cujo desenvolvimento permite um reaproveitamento posterior. Este tipo de prática é cada vez mais comum e assenta em pressupostos que surgiram da chamada “revolução ágil” na década de 90, cujo propósito principal centrava-se na procura pela rapidez de desenvolvimento, sem descurar a eficiência do mesmo (Highsmith, 2004). Ágil também porque permitia metodologias menos rígidas e melhor capazes de reagir à mudança ou outros imprevistos. Por estes motivos, as metodologias ágeis e, particularmente, a prática do desenvolvimento modular podem ser considerados como mais valias para qualquer empresa ou particular na área do desenvolvimento para a Web. Especificamente, este estudo debruçou-se sobre a realidade actual da empresa Dreamlab e procurou investigar

e propor novas soluções para uma maior rapidez e capacidade de resposta para o de desenvolvimento de conteúdos Web dos clientes.

A Dreamlab é uma empresa de desenvolvimento e consultoria em multimédia, sediada em Aveiro, cuja actividade abrange as áreas de vídeo (publicitário, institucional e promocional), 3D, Web (projectos on-line e off-line) e *branding*. A afirmação da empresa assenta na criatividade como factor de diferenciação, na investigação e desenvolvimento tecnológico e na qualidade do serviço, oferecendo as soluções mais adequadas aos pedidos dos seus clientes. A empresa foi criada a 14 de Março de 2001 e desde então que tem crescido todos os anos, quer em numero de recursos humanos, quer em numero de projectos executados, clientes angariados e volume de vendas. Este crescimento sustentado é fruto de uma gestão rigorosa e da capacidade de adaptação aos mercados. Actualmente, a Dreamlab é uma organização que trabalha a comunicação interna e externa dos seus clientes, cujas competências permitem o acompanhamento dos mesmos desde a fase de diagnóstico das necessidades à conceptualização e execução dos meios digitais.

2. Questão de investigação

Como optimizar e agilizar o processo de desenvolvimento de sites Web, no contexto específico da dreamlab?

Sendo a questão de investigação um ponto central em qualquer tipo de investigação, é necessária que esta seja devidamente pensada, analisada e correctamente formulada. A questão de partida tem que ser uma questão clara o suficiente para se perceber de forma imediata e exacta o objectivo da investigação a realizar, mais concretamente, o que "*o investigador (…) procura saber, elucidar, compreender melhor*" (Quivy *et al*, 2008: 32). No caso desta investigação, a questão remete claramente para a optimização de um processo já existente. Deve-se ainda delimitar o universo onde o estudo se vai realizar, neste caso específico o contexto empresarial da Dreamlab. A questão de partida é ainda unívoca e exequível, uma vez que se considera uma questão viável e realista. Finalmente, a questão pode-se ainda considerar pertinente, uma vez que remete para uma problemática actual – a procura da agilidade no desenvolvimento de conteúdos para a Web – através da compreensão de fenómenos observados e uma participação activa por parte do investigador.

3. Finalidade e objectivos

Este estudo teve como finalidade dar um contributo prático numa área cada vez mais relevante no contexto do desenvolvimento para a Web, a agilidade. Mais especificamente, a finalidade desta investigação foi a criação de módulos reutilizáveis para os serviços prestados pela Dreamlab. O facto de se tratar de uma investigação feita em contexto empresarial aponta para um caso particular, no entanto, tendo em conta a escolha dos módulos a desenvolvidos, os resultados obtidos poderão ser mais ou menos generalizados. Também por se tratar do caso específico de uma empresa real, um dos objectivos primários passou pela familiarização com todo o processo de desenvolvimento de serviços por parte da empresa em questão. Foi ainda propósito desta investigação recolher informação acerca de quais os pedidos mais efectuados pelos clientes, pois se o objectivo do estudo passou pela optimização do desenvolvimento para a Web, considerámos que seria mais vantajoso que os protótipos escolhidos para desenvolver representassem alguns dos pedidos mais recorrentes dos clientes. Em síntese, os objectivos primários que foram identificados são os seguintes:

- identificar o processo de desenvolvimento de serviços Web na Dreamlab;
- identificar quais os serviços Web pedidos mais recorrentemente;
- desenvolver um serviço passível de ser vendido como um módulo independente em resposta ao pedido de qualquer cliente.

Atingidos estes objectivos, considerou-se ainda relevante investigar sobre a plasticidade dos módulos desenvolvidos, ou seja, sobre a possibilidade da base do serviço poder ser adaptada para diferentes soluções em contextos distintos. Assim, realça-se ainda como objectivo secundário:

- identificar quais os serviços Web pedidos passíveis de ser adaptados a situações diferentes.

II. Enquadramento teórico

1. Introdução

O desenvolvimento para a Web pode ser compreendido como algo de orgânico, em constante evolução. Esta evolução é alimentada pelos avanços tecnológicos mas também pela necessidade de adaptação ao contexto actual e a necessidades específicas do mercado. Esta investigação recaiu precisamente numa destas necessidades que é a de agilizar o processo de criação de serviços online. Pese embora esta investigação apresentar uma natureza eminentemente prática, foi realizado um levantamento teórico dos conceitos que foram abordados ao logo do processo. Uma vez que este trabalho se insere no âmbito do desenvolvimento de software para a Web, interessa conhecer a evolução do mesmo, ainda que de forma sucinta. Este enquadramento teórico aborda ainda a temática da agilidade, desde o manifesto ágil até às vantagens que esta mudança de mentalidades visa atingir. Já relativamente ao processo de criação de módulos, que representa a parte central desta investigação, este enquadramento pretende definir correctamente os temos usados e faz o levantamento das suas características para uma melhor compreensão de todo o processo descrito.

2. Programação para a Web

a. breve evolução histórica

Desde meados da década de 90 que o desenvolvimento para a Web tem sido das indústrias de maior crescimento por todo o mundo, aumento este que foi ainda ajudado pelo *boom* da Internet durante o ano de 1994. Aproveitando alguns conceitos de sistemas desenvolvidos nas décadas de 40 e 60, o primeiro *browser* gráfico foi desenvolvido em 1991 por Tim Berners-Lee (Grosskurth et al, 2006) e servia de editor de HTML – *HyperText Markup Language*. Documentos formatados desta forma permitem ao autor definir ligações de hipertexto para outros documentos ou diferentes regiões dentro do mesmo documento. Este formato acabaria por se tornar a base do desenvolvimento para a Web e uma norma estandardizada, interpretado do lado dos utilizadores por via de Web *browsers*. Estes fazem pedidos de documentos a servidores remotos, normalmente por HTTP – *HyperText Transfer Protocol* – e dispõem a informação visualmente no monitor.

Devido à sua relativa simplicidade e limitações tecnológicas, surgem mais tarde outras tecnologias que servem de complemento ao HTML para melhorar o aspecto visual dos documentos (Grosskurth et al, 2006). A primeira e mais usada são os CSS – *Cascading Style Sheets*. A principal vantagem destes CSS passa pela inclusão de informação de *layout* e estilos (cores e fontes, por exemplo), cuja aplicação é simples e não complica a estrutura original do documento. Uma primeira proposta para o uso destas folhas de estilo surgiu em 1994, apesar de só 2 anos mais tarde sair uma versão final. A versão original foi lançada em 1996 e está a ser desenvolvida actualmente uma terceira evolução, há já mais de 11 anos. No mesmo ano do lançamento da primeira versão dos CSS a Microsoft lança o Internet Explorer 3, no entanto o suporte para CSS era ainda limitado, e assim se manteve durante mais de 3 anos, até ao lançamento do Internet Explorer 5.0, em 2000, que já suportava as folhas de estilos quase na sua totalidade (Grosskurth et al, 2006). Em 1995, surge a tecnologia JavaScript, cuja aplicação no código HTML e consequente interpretação aplica efeitos nos conteúdos, de outra forma estáticos provenientes de documentos HTML. Estes efeitos são regularmente mudança de foco nos elementos e interpretação de acções despoletadas pelo rato. JavaScript é uma linguagem de script orientada a objectos e funciona do lado do cliente. Apesar de ter sido influenciada por várias linguagens de programação, foi desenhada para se aproximar à linguagem Java (sem relação com JavaScript) apenas com o objectivo de se tornar mais acessível (Grosskurth et al, 2006). Também em 1995 surgem as primeiras aplicações em Java que, tal como conteúdos Flash (a partir de 1996), implicam a instalação de *players* próprios para correcto funcionamento nos *browsers*. Estas aplicações embebidas nas páginas aumentam as potencialidades do desenvolvimento para a Web exponencialmente, conferindo características que de outra maneira não seriam possíveis em páginas normais. É agora possível criar documentos com base em informação visual vectorial, adaptando-se assim a diferentes utilizadores e resoluções sem perder qualidade. Com base nestas novas potencialidades começam a ser desenvolvidas RIAs para a Web – *Rich Internet Aplication*, ou seja, aplicações cujas aparência e funcionalidades estão cada vez mais próximas a aplicações *desktop*. Estas aplicações transferem o processamento necessário para a interface final para o cliente Web, e mantêm a maioria da informação no lado do servidor. Esta constante mudança de tecnologia usada é um reflexo das diferentes necessidades de dinamismo e interactividade por parte das páginas Web (Junia Cristina et al, 2007). Esta preocupação pela interactividade aponta para uma procura de formas de agilizar a navegação dentro de blocos de informação.

Podemos, pois, dividir o desenvolvimento para Web em *client side*, *server side* ou misto. *Client side* remete para tecnologias como o Flash, JavaScript e Ajax, onde o processamento é feito do lado do cliente, enquanto que no caso das tecnologias *server side* o processamento principal é feito remotamente no servidor e exemplo disso são as tecnologias ASP, Java e PHP.

Também a mudança de paradigma de Web implica toda uma mudança de mentalidades no desenvolvimento de conteúdos on-line (Junia Cristina et al, 2007). O aspecto técnico do desenvolvimento para a Web tem evoluído de forma paralela a esta mudança de mentalidades, enquanto se passa de uma Web 1.0 centrada na informação, onde o utilizador é apenas um consumidor, para um conceito de Web 2.0 (termo criado por Tim O'Reilly em 2006 no seu blog[1]), onde o utilizador é agora o consumidor e produtor de conteúdos. Actualmente surge o conceito da Web social onde o consumidor assume o papel central de consumidor, produtor e editor de conteúdos.

É ainda de referir que actualmente o desenvolvimento de conteúdos para a Web não é restringido apenas ao computador e monitor, uma vez que se tem verificado um crescimento de navegação e consulta por via de dispositivos móveis. Este aumento exponencial de potenciais clientes finais em cenários multiplataforma implica constantes mudanças ao nível da mentalidade de desenvolvimento e design. Também o aumento do número de ferramentas de autoria indica um cada vez maior, mais fácil e rápido desenvolvimento de conteúdos para a Web.

b. definição de conceitos

A ***arquitectura de um software*** é a estrutura dos componentes interligados através de uma ou mais interfaces (Bass et al, 2003). Na arquitectura são ainda descritos os princípios e guias acerca do design e evolução no tempo. Estes componentes podem ser compostos por componentes menores e respectivas interfaces para interacção entre ambos. As arquitecturas de software estruturadas têm como objectivo permitir uma visão abstracta de todos os módulos e funções/procedimentos e regem-se pelos seguintes princípios: decomposição modular, informação escondida e independência funcional.

O conceito de "dividir para conquistar" está intimamente ligado ao objectivo desta investigação, remetendo assim para a identificação e divisão dos serviços em funcionalidades autónomas (Bass et al, 2003).

[1] what is web 2.0, http://oreilly.com/web2/archive/what-is-web-20.html (15-06-2010)

A informação escondida aponta para a necessidade que há de evitar que os dados sejam acessíveis entre os módulos, evitando assim problemas em caso de alterações posteriores nos mesmos módulos (Bass et al, 2003).

Por fim, a independência funcional de um módulo mede-se através de dois critérios: a coesão, que é a ligação existente entre os vários módulos e quanto maior, melhor, e o acoplamento que remete para o grau de interdependência entre os módulos e quer-se o menor possível, em condições óptimas (Bass et al. 2003).

Estes princípios são conseguidos principalmente através de uma decomposição funcional do problema em questão e um refinamento sucessivo passo-a-passo, apontando para subsistemas cada vez mais pequenos e objectivos. O ***processo de design arquitectural*** divide-se, pois, em três partes: a estruturação do sistema, onde este é composto em vários subsistemas e é identificada a comunicação ente eles; a modelação do controlo, onde se cria um modelo dos relacionamentos entre os diferentes subsistemas; e, finalmente, a decomposição modular onde os subsistemas são decompostos em módulos (Bass et al, 2003). Como já foi referido, os subsistemas são independentes dos serviços prestados por outros subsistemas enquanto que os módulos são componentes de um sistema que fornecem conteúdos a outros componentes, mas não são considerados sistemas separados (Sommerville, 2006). A decomposição modular pode correr de duas modalidades distintas: na primeira, o sistema pode ser decomposto em objectos em interacção; na segunda, é utilizado um modelo de *pipeline* ou modelo de fluxo, onde o sistema é decomposto em módulos funcionais que transformam entradas em saídas. O modelo baseado em objectos tem como principal vantagem a fraca acoplagem dos objectos, permitindo assim a edição dos mesmos em afectar outros objectos. Já o modelo de *pipeline* é relativamente fácil de implementar como modelo concorrente ou sequencial (Sommerville, 2006).

O ***paradigma de programação*** de um programa é o conjunto de formas que servem de padrão a seguir; funciona como uma escola de pensamento, tendo em vista a solução de problema específico (Silva et al, s.d.). Dentro da programação há quatro paradigmas principais, o imperativo, o funcional, o lógico e o orientado a objectos, sendo que é possível utilizar mais que um paradigma. O paradigma imperativo remete para um tipo de programação procedimental rígida (do género faz “A e depois B”) onde os passos - comandos - têm uma ordem fixa para o correcto funcionamento. O paradigma funcional tem por base a teoria das funções matemáticas, onde se avalia primeiro uma expressão e depois se usa o valor resultante para algo. O paradigma lógico é o que parece menos natural e baseia-se na resposta de uma pergunta através da procura por uma solução, ou

seja, “para conseguirmos o resultado C então A e B”. O paradigma de programação por objectos é actualmente o mais popular, em parte devido ao encapsulamento que permite, um factor de grande peso quando os programas começam a aumentar exponencial. Neste paradigma são enviadas mensagens entre objectos distintos para simular a evolução temporal de um fenómeno real (Silva et al, s.d.).

Finalmente, a ***metodologia de desenvolvimento de software*** remete para a estrutura conceptual usada para estruturar, planear e controlar o processo de desenvolvimento de um produto (Cockburn, 2006). As principais metodologias usadas para o desenvolvimento de conteúdos para a Web são o modelo em cascata, em espiral, iterativo (Gaedke et al, 2000), *cowboy coding* e desenvolvimento ágil. A metodologia tem uma estrutura linear sequencial e como a fase de planeamento só ocorre uma vez, não se define a resposta a qualquer resultado inesperado intermédio. A metodologia em espiral baseia-se em camadas e inclui a criação de protótipos para testar se o projecto está a decorrer bem antes de passar às fases seguintes; no entanto, o decorrer das fases é também linear. O modelo iterativo melhora sobre o modelo em espiral e permite testar os subsistemas nas iterações iniciais e permite assim melhorar a flexibilidade do processo; no entanto o processo base continua a ser linear. Contrastante com estas metodologias, surge o *cowboy coding* que é precisamente a ausência de um método definido: os elementos da equipa fazem o querem quando quererem. Finalmente surgem as metodologias ágeis, cujo objectivo é encorajar e suportar a flexibilidade, conferindo assim um elevado grau de tolerância à mudança a todo o processo. Exemplo disso são por exemplo as metodologias SCRUM, Crystal ou *Extreme Programing* (Cockburn, 2006).

3. Agilidade

a. revolução ágil

Desenvolvidas durante a década de 90, as metodologias adaptáveis foram criadas por e para *developers*, em forma de reacção às metodologias lineares (Highsmith, 2004). Esta prática tem como particularidade adaptar-se às necessidades do projecto em mão, de forma mais rápida e flexível (ágil), mesmo quando as especificações deste mudam durante o seu desenvolvimento, permitindo perder o mínimo de trabalho e recursos possíveis. O desenvolvimento ágil segue de perto o valor do negócio, através de actividades que contribuem de forma directa para o objectivo final do projecto (Cockburn,

2006). Estas metodologias foram criadas especialmente para promover e facilitar a comunicação, colaboração e coordenação dentro de equipas dinâmicas, normalmente compostas por grupos pequenos. Em 2001 foi criado o manifesto ágil pela Agile Alliance, composta por 17 elementos influentes na área, que sumariza toda a mentalidade do desenvolvimento ágil (Whitworth, 2006) e é o seguinte:

"We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more."

É assim evidente a preocupação em centralizar os esforços das equipas na comunicação e colaboração interpessoal e, acima de tudo, o estar sempre preparado a eventuais mudanças e não seguir um plano rígido (Cockburn, 2006), não querendo com isto dizer que todas as práticas anteriores estão completamente erradas e/ou desactualizadas. O manifesto ágil estabeleceu assim os valores principais nos quais todas as metodologias ágeis se viriam a basear, não pela criação de um processo novo, mas pela alteração de mentalidades (Highsmith, 2004). É ainda de realçar que, apesar de no manifesto falar especificamente em software, este modelo pode ser aplicável a outras áreas.

b. visão e inovação

O autor Jim Highsmith (2004) refere a existência de uma revolução silenciosa por parte dos produtores de conteúdos, num mercado que exige cada vez mais inovação, mais rapidamente e com menores custos. Assim, o autor aponta para um conjunto de objectivos cruciais no desenvolvimento ágil, sendo o primeiro a procura contínua por inovação, algo que não se consegue em ambientes rígidos, mas apenas com base em culturas adaptáveis, valorizando princípios como a auto-organização e auto-disciplina.

Outro objectivo a ter em vista é a adaptabilidade dos produtos, funcionando assim como uma salvaguarda para as constantes mudanças a que os conteúdos estarão sujeitos, ao nível do mercado, tecnologias ou qualquer pedido em particular, permitindo assim que este se mantenha actual e relevante mesmo com o passar do tempo. Também as pessoas envolvidas no desenvolvimento destes produtos têm que ser adaptáveis às mudanças anteriormente referidas, sempre que necessário. Para isso é preciso que os membros das equipas tenham uma mentalidade que veja a mudança não como um obstáculo, mas como um desafio dinâmico. Outro objectivo é transversal a qualquer tipo de negócio, a redução dos prazos de entrega dos produtos, principalmente por prestar uma constante atenção às especificações do produto final e qual será a melhor equipa para o desenvolver. O objectivo final aponta para a garantia de obtenção de resultados que cumpram sempre os pedidos dos clientes, dentro dos constrangimentos temporais e monetários.

Um dos principais valores da agilidade é a resposta à mudança, em vez de seguir um plano rígido. Todos os projectos têm certezas e incertezas e para obter resultados positivos é necessário um balanço entre planeamento e mudança. Assim, considera-se que deve ser favorecida a exploração em vez de um plano de tarefas. Há ainda que ter em conta o custo das iterações projectuais, sendo sempre preferível um modelo adaptativo onde todo o processo evolui de forma paralela ao produto final. Esta adaptação ocorre também no planeamento, onde em vez de se fazer uma previsão do calendário nas fases iniciais, este vai se adaptando com o desenrolar do processo de desenvolvimento. Outro valor é a preferência dos produtos pela documentação, que aponta para o teste dos produtos em contexto real (Paige et al, 2002). Por melhor que seja a documentação de um projecto, o produto final só pode ser testado depois de devidamente produzido e testado junto do público-alvo. Uma boa explicação não é substituto da interacção. Só assim se consegue recolher o feedback que realmente interessa e se percebe o grau de sucesso. Tal como tudo no manifesto ágil, este parâmetro não quer dizer que a documentação é desnecessária, apenas que é menos importante que uma versão funcional do produto em questão (Highsmith, 2004) (Cockburn, 2006). Os restantes valores relacionam-se com a comunicação e interacção entre todos os envolvidos no processo de criação de produtos. Relativamente ao clientes, é necessário assegurar um ambiente colaborativo que não se restrinja à assinatura de um contracto. O cliente define os atributos do produto naquele momento, mas o verdadeiro valor está na adaptabilidade dos produtos face ao seu futuro. Finalmente, a agilidade depende de pessoas que consigam fazer aquilo que todas as ferramentas do

mundo não conseguem. Sem retirar valor à necessidade de ferramentas correctas para a tarefa em mãos, as decisões e análises continuam a recair sobre as pessoas e as suas capacidades. É importante que estas pessoas, únicas e dotadas dos talentos necessários, se mantenham em constante contacto entre si e que os processos se adaptem às pessoas, e não o contrário.

c. modelo de desenvolvimento ágil

Um modelo de desenvolvimento deve reflectir objectivamente o produto final e, como tal, se o objectivo é a criação de algo inovador, o modelo deve ser flexível (Highsmith, 2004). Essa flexibilidade passa por reflectir e dar ênfase aos valores presentes no manifesto ágil, anteriormente descritos. O modelo de desenvolvimento ágil baseia-se em cinco passos, cujos nomes descrevem sumariamente o objectivo de cada fase.

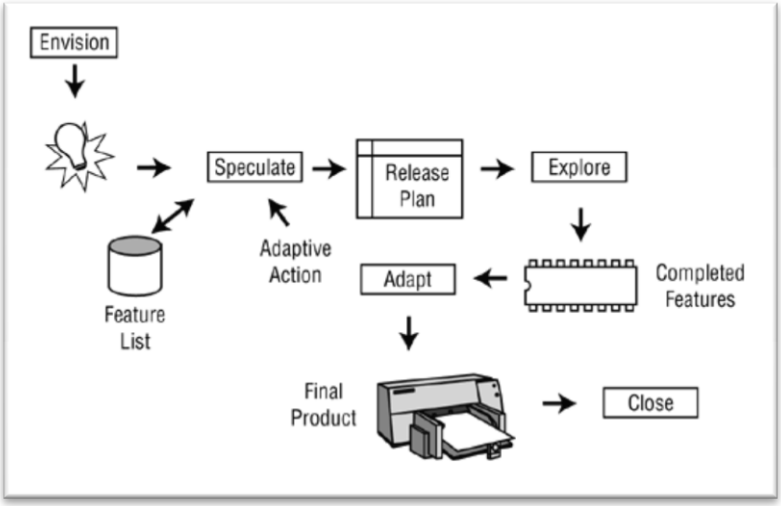

Ilustração 1 - Framework de desenvolvimento ágil (Highsmith, 2005)

A primeira fase é a de visionamento, onde se define o factor crítico de sucesso do produto e se aborda o “o quê”, “quem” e “como” do projecto. Estas perguntas são feitas por esta mesma ordem, sendo que a última reflecte a preocupação de perceber como vão os elementos da equipa trabalhar em conjunto.

Na fase seguinte, especula-se sobre o projecto em mãos, mesmo ainda não tendo todas as informações necessárias. Começam-se a definir os requisitos do produto, cria-se um plano de entregas e estratégias para lidar com o risco do projecto. É também nesta fase que se começam a fazer estimativas financeiras.

A próxima fase é a de exploração, onde se cria um grupo projectual colaborativo e responsável. É ainda nesta fase que se gerem as interacções da equipa com os clientes e outros elementos envolvidos.

Na quarta fase termina o ciclo que se vai repetindo ao longo do desenvolvimento do projecto. Este ciclo começa na fase de especulação e termina com a fase de adaptação, onde se revêem os resultados por todos os pontos de vista possíveis e se melhora o produto cada vez que ocorre.

A quinta fase consiste no fecho do projecto, onde o objectivo principal é a aprendizagem e o aproveitamento dessa aprendizagem para uma próxima equipa projectual. Da mesma maneira que demasiada linearidade num modelo eventualmente leva à estagnação, também a procura desmesurada de evolução não é particularmente positiva, podendo levar a mudanças injustificadas e constantes; ficando assim a cargo do julgamento dos responsáveis por cada projecto conseguir encontrar um equilíbrio no modelo usado, sempre com vista ao objectivo final.

4. Arquitectura Orientada a Serviços

a. o que é

A arquitectura orientada a serviços – SOA, é um paradigma de organização e utilização de capacidades com diferentes autorias (Reitman, 2007). Nos SOA há um enfoque nos serviços como sendo uma necessidade passível de ser transmitida a terceiros, ou seja, uma entidade cria uma aplicação para resolução de um problema especifico e negoceia essa solução com terceiros (Arsanjani, 2004). Esta abordagem tem vindo a ser cada vez mais utilizada graças a uma promessa de reutilização dos componentes, maior flexibilidade e redução da complexidade dos sistemas que a compõem (Windley, 2006). O principal objectivo da aplicação deste tipo de arquitecturas é o aumento da eficiência, agilidade e produtividade, quando comparadas com outros paradigmas diferentes. A utilização destes serviços não é necessariamente efectuada numa relação de um para um, entre necessidades e capacidades, uma vez que a granularidade – característica que aponta para a especificidade de cada serviço para a resolução de um problema – pode variar e com isso implicar a combinação de mais de uma capacidade, ao mesmo tempo que um mesmo serviço pode ser aplicado a mais do que uma necessidade (Windley, 2006).

i. princípios gerais

A correcta utilização desta metodologia baseia-se em alguns princípios que importa assegurar (Erl, 2005), como é o caso da ***reutilização***, ou seja, todos os serviços disponibilizados podem potencialmente ser reutilizados para outros casos. Este é o princípio base dos SOA. Como já foi referido anteriormente, há ainda que ter em conta a granularidade dos serviços disponibilizados, ou seja, a especificidade dos mesmos, tendo em vista a resolução de um problema.

Outro principio é o da ***modularidade***, ou seja, garantir que cada serviço é auto-suficiente e funciona de forma autónoma, independentemente das condições da sua implementação. Também a "composabilidade" e "componentização" são factores a ter em conta nesta arquitectura, apontando estes para a possibilidade de suportar a composição de serviços mais complexos através da utilização destes mesmos serviços como componentes encadeados numa sequencia específica (Erl, 2005). Uma vez que, na sua raiz, este modelo baseia-se na disponibilização de serviços, é ainda de sublinhar a importância da aderência a standards necessária, permitindo assim maximizar as potencialidades de interoperabilidade entre serviços. Também com vista a maximizar a eficiência deste tipo de serviços é necessária a correcta identificação e categorização dos mesmos, de forma a facilitar a pesquisa e retorno do serviço correcto para cada cenário específico (Erl, 2005).

Outro principio inerente aos SOA é o de ***encapsulamento***, que determina que cada serviço é visto como uma "caixa negra" onde, para o utilizador final, apenas é necessário saber quais os dados de entrada e o resultado de saída. Aliada a esta particularidade está também a separação da camada de negócio da tecnologia de base: com efeito, a abstracção dos serviços permite que em qualquer altura do ciclo de vida do serviço a tecnologia base seja alterada e a lógica de negócio se mantenha estável. Uma vez que os serviços têm como finalidade a reutilização em cenários de maior complexidade, estes devem ser implementados apenas uma vez – implementação única – no meio onde serão utilizados (Erl, 2005).

Outro princípio base dos SOA é o ***reaproveitamento*** das lógicas existentes: deve-se, sempre que for possível, adaptar serviços com apenas uma aplicação para serviços inter-operáveis. Os serviços dependem de si para funcionar; no entanto, para garantir a comunicação entre si existem manifestos ou contractos que indicam os parâmetros de entrada e valores de retorno final de cada execução para cada serviço. Os

serviços disponibilizados devem ainda ter em conta questões como o desempenho e escalabilidade da operação para casos de utilização mais avançada – uso eficiente dos recursos de sistema. É ainda importante que a implementação dos serviços só ocorra quando estes já possuem a maturidade necessária para garantir a robustez e desempenho necessários. Finalmente, é ainda de referir que, tal como qualquer outro sistema de produção, é necessário gerir o ciclo de vida de um serviço (Ali Arsanjani, 2004), algo que será abordado mais à frente no documento.

ii. efeito de caixa negra

Como já foi referido anteriormente nos SOA um serviço funciona como uma unidade autónoma independente da plataforma, podendo ser publicado e agregado a outros serviços para satisfazer a função para a qual foi originalmente desenhado. Estes serviços podem ser divididos em duas categorias: *data services*, serviços que permitem pesquisar e editar informação e *decision services*, serviços de tomada de decisão consoante o tipo de dados que recebem. Para efeitos de encapsulamento, os serviços são desenvolvidos de maneira a que, para o cliente final, apenas seja necessário saber quais os parâmetros de entrada e os valores de retorno, abstraindo-se completamente do que se passa na camada técnica do serviço, efeito este ao qual se dá o nome de efeito caixa negra (Hsuan, 1998). Esta abstracção do serviço ajuda ainda na reutilização dos serviços, onde todos os detalhes são escondidos dos consumidores, estando apenas visível um interface publico genérico. Este efeito é ainda vantajoso quando ocorrem alterações no serviço na medida em que, mantendo os parâmetros de entrada é possível substituir o serviço sem directamente, sem quaisquer tipo de implicações para o consumidor do serviço. Também do ponto de vista de desenvolvimento dos serviços estes podem ser considerados como uma caixa negra, o gestor de negócio é apenas responsável pelo processo de negócio e coordenação de actividades com os restantes membros do departamento de informática, que vão desenvolver o serviço. Estes arquitectos, designers e *developers* são todos responsáveis pelo serviço até certo ponto; no entanto, se o serviço não funcionar é o gestor de negócio que é responsabilizado, mesmo não tendo ideia do que se passa no serviço, do ponto de vista tecnológico.

b. como funciona

Tendo em conta a natureza da prestação do serviço em causa, são necessárias garantias de confiança e não repudiação. Isto é, na prestação de serviços, a "empresa A" confia na informação conseguida pela "aplicação B", da mesma forma que nesta mesma prestação a "aplicação B" é responsável pela informação e garante a autenticação perante o serviço, podendo assim registar o mesmo (Windley, 2006). Uma vez que a base dos SOA é a prestação de um serviço em resposta a uma necessidade específica, é possível definir os intervenientes como entidades com interesse no resultado do uso de um serviço e simultaneamente dividi-los em participantes e não participantes.

Definem-se como participantes os intervenientes que interagem no processo e o seu interesse reside no uso bem sucedido dos serviços. Por outro lado, os não participantes são os intervenientes que não interagem no processo sendo, no entanto, afectados pelo uso de recursos e podem ter interesse no resultado do processo (Windley, 2006).

Finalmente, do ponto de vista do processo de negócio definem-se ainda três tipos de intervenientes participantes e são eles: o fornecedor de serviço, o consumidor de serviço e os agentes. Fornecedor, tal como o nome indica, é o participante que disponibiliza o serviço para o consumidor de serviços, ou seja, o participante que precisa do serviço com um determinado objectivo (Rotem-Gal-Oz, s.d.). Assim, os agentes são as entidades necessárias que agem em nome de uma empresa ou organização de modo a permitir o consumo de um serviço (tipicamente aplicações de software e dispositivos de hardware) (Windley, 2006).

c. condições ideais

Tal como o autor Thomas Erl (2005) afirma, os ideais representam um estado de excelência que nos motiva a atingir estados mais além do que seria possível sem que houvessem ideais para nos guiarmos. Assim a orientação a serviços representa uma visão ideal onde todos os recursos são divididos e usados consoante as necessidades. Quando aplicada às tecnologias de informação, a orientação a serviços cria um modelo de automatização lógica, quando apoiada no conceito de abstracção, permitindo assim uma globalização dos serviços. Esta abstracção sobrepõe-se a qualquer tipo de dificuldades técnicas e, devido à adesão a standards, permite que os serviços sejam aproveitados de uma forma muito mais massificada e ao mesmo tempo simples para os

consumidores do serviço. Esta visão ideal fez que a orientação a serviços seja actualmente considerada como o próximo passo na automatização de negócios, assumindo-se como sucessor das tradicionais arquitecturas distribuídas.

d. soa governance

SOA Governance é um conceito usado no que refere ao controlo sobre serviços disponibilizados em caso de arquitecturas orientadas a serviços. Esse controlo tem em vista a ajuda nas tomadas de decisões, e melhor definição dos papeis e responsabilidades necessárias para a obtenção do sucesso esperado por qualquer entidade (Erl, 2005). O Instituto Gartner define SOA Governance como o processo de assegurar e validar que todas as "habilidades" e artefactos dentro de uma arquitectura estão a funcionar como seria de esperar e mantêm um determinado nível de qualidade (Natis, 2003). Tipicamente, alguns aspectos deste controlo recaem sobre a garantia de adesão a standards, o assegurar a qualidade de serviços e a gestão de mudança. Ou seja, exercer controlo quando um serviço precisa de ser alterado, analisar o tipo de implicações que isso poderá trazer ao consumidor e, finalmente, garantir o retorno do valor aos accionistas, momento este que é esperado em qualquer negócio, seja qual for o modelo referencial implementado.

Faz ainda parte desta mesma gestão criar e saber gerir um portfolio de serviços disponíveis, que passa ainda pela criação de serviços novos e manter actualizados os já existentes (Windley, 2006). É também necessária a gestão do ciclo de vida dos serviços, tendo sempre em conta que, quem está a consumir os serviços, não pode ser prejudicado em nenhum momento. Esta gestão do ciclo de vida implica ainda um constante acompanhamento dos serviços e supervisão da performance dos mesmos (Windley, 2006).

e. web services

Actualmente o uso mais recorrente do modelo SOA é sob a forma de web-services (Rotem-Gal-Oz, s.d.) – definidos no glossário W3C como aplicações de software desenvolvidas com vista à interoperabilidade em rede, através de protocolos standardizados[2]. Estes vêm encapsulados em envelopes SOAP – *simple object access*

[2] web services glossary, http://www.w3.org/TR/ws-gloss/ (15-05-2010)

protocol – e são comunicados via protocolos HTTP ou outros e são baseados em esquemas standardizados XML – *extensible markup language* – uma norma de estruturação de informação em documentos, tipicamente textual com informação e meta-informação ("informação sobre a informação") no mesmo documento. Finalmente a definição dos contratos para prestação dos serviços surge em WSDL – *web service definition language* – onde se especificam os métodos usados, os argumentos de entrada e os valores de retorno. Em termos gerais, o WSDL explica o funcionamento da caixa negra e é o contracto que vincula o consumidor e fornecedor do serviço (Erl, 2005).

5. Desenvolvimento Orientado a Aspectos

a. o que é

O desenvolvimento de software orientado a aspectos - AOSD - ou simplesmente a "programação orientada a aspectos" é uma metodologia emergente no desenvolvimento de software que visa a modularização e resolução de problemas que esta levanta em abordagens como a programação estruturada e a programação orientada a objectos, às quais serve de complemento (Wampler, 2003). O objectivo máximo desta metodologia passa por restaurar a modularidade dos *concerns* que utiliza. Para isso é necessária uma correcta identificação dos *crosscutting concerns* e extraí-los como módulos separados, chamados de *aspects*, passíveis de ser desenvolvidos e geridos de forma independente. O objectivo final passa ainda pela composição conjunta dos módulos para criação de sistemas completos e funcionais (Soares, 2004).

Como foi referido inicialmente, trata-se ainda de uma metodologia emergente e como tal ainda não é usada de forma massificada e com base nisso mesmo, o autor Dean Wampler (2003: 15) faz a analogia "*Object Oriented Programming went mainstream when its modularity capabilities became essencial, Aspect Oriented Programming will go mainstream when its modularity capabilities become essencial*", apontando também para o então actual estado do mercado relativamente ao desenvolvimento de software e para a aquilo que considerava ser uma imaturidade da metodologia, mas faz ainda a ressalva "*but only when tools and processes are ready!*" remetendo assim para o aspecto mais técnico da implementação.

b. como funciona

Esta metodologia visa a implementação de serviços de forma modular, com enfoque em *concerns*[3] ou seja áreas de interesse dentro do sistema. Estas áreas são um objectivo ou funcionalidade específica, que funciona como critério de decomposição do software em partes mais pequenas e cuja correcta identificação e separação permitem a criação de aspectos (Wampler, 2003). Estes aspectos poderão ser depois associados ao conceito de *crosscutting concerns*, que aponta para uma relação entre áreas de interesse onde um aspecto do programa afecta outros, numa relação tipicamente hierárquica. Uma das motivações da programação orientada a aspectos é a capacidade de modularização destas áreas de interesse, algo que se não é correctamente implementado pode levar a que o código esteja "espalhado" ou disperso (*scattered*), ou seja, a sua representação (invocação do serviço) encontra-se em diversos módulos ou então *tangled* quando o código está misturado com código referente à implementação de outros *concerns*. Este código "espalhado e misturado" acaba por dificultar no desenvolvimento e manutenção do produto, devido a uma deficiente organização do código e incorrecta identificação dos módulos. Outra das especificidades do AOSD é que cada funcionalidade particular de um programa é um aspecto e funciona em conjunto com os restantes aspectos através de uma característica a que se dá o nome de "tecelagem" (*weaving* ou composição). Esta composição dos aspectos todos que compõe um produto pode acontecer durante a compilação do código – *static weaving* – ou então durante o carregamento da aplicação – *dynamic weaving* (Jackobson et al, 2004). Como já foi referido, esta metodologia tem como objectivo a resolução de problemas que possam surgir relativamente à modularidade, processo que a Aspect-Oriented Software Association define como a separação destes *concerns*.

O termo modularidade usa-se para referência ao nível de separação clara entre unidades individuais – os módulos. Considera-se um *concern* modular quando o texto relativo ao código está local presente localmente, quando há uma interface que define correctamente a interacção com o restante sistema e ao mesmo tempo é uma abstracção da implementação (conceito anteriormente abordado), quando existe um mecanismo automático que garante a satisfação do serviço de cada módulo e correcto funcionamento em conjunto com as restantes interfaces. Os módulos têm ainda que ser compostos automaticamente por compiladores, *loaders* ou outros, em variadas configurações em conjunto com outros módulos para formar sistemas completos (Wampler, 2003).

[3] AOSD glossary, http://www.aosd.net/wiki/index.php?title=Glossary (15-05-2010)

c. separação de assuntos

A separação de assuntos está intimamente ligada à criação de módulos e surge como resposta a um problema no desenvolvimento de software: o desenvolvimento de várias tarefas ao mesmo tempo potencia o surgimento de erros. Esta ocorrência deve-se à elevada complexidade do processo de desenvolvimento de software que engloba, de forma geral, várias tarefas, objectivos e delegação de diferentes responsabilidades. Assim, a separação de assuntos permite a divisão de um problema em partes que possam ser realizadas de forma individual e posteriormente criar um linha de entendimento na forma como as partes irão depender, interagir e relacionar-se umas com as outras (Wampler, 2003).

Esta separação pode ser feita tendo em conta cinco aspectos diferentes: a separação temporal, de qualidades, de vistas, de partes e, finalmente, de responsabilidades. A separação temporal aponta para um planeamento agendado de forma reduzir a comutação entre actividades. A separação das qualidades aponta para o desenvolvimento de um programa por funcionalidades (Jackobson et al, 2004). A separação de vistas remete para a concentração no fluxo de dados ou fluxo de controlo. A separação de partes de um sistema ocorre geralmente em sistemas muito complexos e remete para uma divisão em termos de tamanho. Finalmente, a separação de responsabilidades, tal como o próprio nome indica aponta para divisão do trabalho dentro de uma equipa, cada elemento com diferentes tarefas e responsabilidades (Jackobson et al, 2004).

6. Modularidade

a. como surgiu

A modularidade é um conceito que não se restringe ao universo da engenharia de software e remete para o processo cognitivo ou perceptivo de um todo em sub-processos. Actualmente a maioria dos processos e produtos de engenharia são modulares, como por exemplo, as linhas de montagens de carros. O autor Carliss Young Baldwin (2007) define a modularidade como um padrão específico de dependências entre

elementos de um sistema. Este conceito foi popularizado inicialmente pelos romanos com a frase em latim *divide et impera*, dividir para conquistar. Este pequeno lema sublinha assim a separação de assuntos: qualquer um problema maior pode ser mais facilmente resolvido quando dividido em problemas mais específicos e de menor dimensão (Baldwin et al, 1999). Tipicamente o objectivo da resolução destes problemas menores é depois combinado para criação da solução para o problema inicial, de maiores dimensões. Deste processo de divisão ocorre a criação de módulos, que são processos cognitivos ou perceptivos que funcionam com relativa independência da restante arquitectura cognitiva, podendo assim ser analisados e compreendidos com um certo nível de independência do sistema total onde se inserem (McClamrock, s.d.). Relativamente aos sistemas, diz-se modular quando um dado sistema pode ser decomposto num determinado número de componentes menores, que interagem e trocam recursos entre si através de interfaces estandardizadas (Baldwin et al, 1999).

b. objectivos gerais

A procura pela modularidade tem três objectivos principais que se interligam: a "decomponibilidade", a "componibilidade" e, finalmente, a compreensão. Decomponibilidade remete para a divisão de um problema em subproblemas e decompor cada um dos sub-problemas até que estes desempenhem uma só função especifica no âmbito de um problema maior (Baldwin et al, 1997). Depois de divididos os componentes, surge a "componibilidade", ou seja, a integração de todos os componentes elementares até se obter um sistema completo. Como já foi referido anteriormente, esta integração e posterior interacção entre os módulos só é conseguida através de interfaces standardizados no que se refere à entrada e saída de informação. Finalmente, outro objectivo do trabalho com módulos é uma maior facilidade de compreensão e identificação do código para determinados fins. Um objectivo secundário passa pela ocultação da informação, que dita que esta não deve estar disponível para terceiros, sempre que não for estritamente necessária (Baldwin, et al 1997).

c. como funciona

O conceito da modularidade remete para um conjunto de condições ideais onde todas as aplicações seriam tipicamente desenvolvidas com base na integração de componentes reutilizáveis. O funcionamento de um sistema modular e a obtenção de

“componibilidade”, “decomponibilidade” e “compreensibilidade” depende ainda de dois factores: a coesão e acoplamento entre os módulos (Baldwin et al, 1997). Estes dois critérios são usados para medir a independência funcional. A coesão remete para a ligação existente entre os elementos de um módulo e espera-se alta num sistema eficiente. Já o acoplamento é tanto maior quanto a interdependência entre os módulos de um sistema, sendo assim ideal obter baixos níveis de acoplamento num sistema funcional. Os módulo devem ser projectados de forma a que toda a informação que contêm (procedimentos e dados) esteja sempre inacessível a outros módulos que não tenham necessidade nenhuma nela. Os módulos são aplicados com uma interface modular, ou seja, uma parte externamente visível do módulo através da qual ele comunica com o meio externo. Para compreender o funcionamento de um sistema modular é ainda necessário compreender e implementar de forma correcta conceitos de hierarquia, abstracção e generalidade. Outro conceito de elevada importância dentro do âmbito da modularidade é o de separação de assuntos, que já foi abordado no capítulo anterior (Baldwin et al, 1997).

i. hierarquia

A hierarquia de controlo de um sistema representa a organização dos componentes que integram uma dada aplicação. Esta representação gráfica não reflecte aspectos procedimentais da aplicação como tomadas de decisão ou estruturas de repetição (Fernandes et al, s.d.). O uso mais comum recai sobre os diagramas em árvore, nos quais se representa a hierarquia de um sistema de forma arborescente. Estas arborescências reflectem questões como a profundidade de um sistema, que indica o numero de níveis de controlo; a largura, que indica o espaço controlo global; o *fan-out*, que indica o numero de módulos que são directamente controlados por um determinado módulo e o *fan-in* que indica precisamente o contrário, ou seja, quantos módulos controlam directamente um determinado módulo. Estruturalmente os módulos podem ser: sequenciais, quando são chamados e executados sem interrupções; incrementais, quando podem ser interrompidos antes da sua conclusão e subsequentemente reiniciado a partir do ponto de interrupção; e, finalmente, módulos paralelos, quando executados simultaneamente com outros módulos em ambientes de multiprocessamento concorrente (Batory et al, 1992).

ii. abstracção

A abstracção é o processo que visa a identificação dos aspectos importantes de um determinado acontecimento, em detrimento de tudo o que possa ser considerado acessório (Fernandes et al, s.d.). Esta técnica é usada em todos os campos da engenharia, não se limitando ao contexto desta investigação, de forma a dominar a complexidade inerente a um determinado sistema. Um exemplo disto são todas as equações matemáticas que definem modelos físicos reais. Relativamente a este processo, é ainda de referir que uma mesma realidade pode ter várias formas de abstracção, mediante o seu propósito funcional ou, por outras palavras e pegando no exemplo anterior, um mesmo modelo físico pode ser representado por um gráfico ou por uma equação numérica. No âmbito desta investigação, a procura da abstracção passou pela identificação e distinção do que era realmente necessário do acessório, de forma a poder haver a decomponibilidade de um determinado sistema.

iii. generalidade

A generalidade é uma característica que, dentro do âmbito do engenharia de software, na sua ausência leva ao desenvolvimento de soluções semelhantes. Isto acontece porque durante o processo de desenvolvimento de uma aplicação pode ocorrer a construção de muitos componentes semelhantes ou até mesmo várias vezes. Este tipo de ocorrências tem como impacto imediato o aumento da complexidade geral do sistema. Uma maior atenção à generalidade de funções permite o aumento da reutilização (Ribeiro, 2008), quer por parte de outros programas, quer do mesmo programas em fases diferentes. Uma elevada generalidade permite ainda um aumento da robustez geral do sistema e também um desenvolvimento mais rápido do mesmo, devido à criação única de rotinas que são utilizadas sempre que o mesmo problema surgir. A robustez está ligada ao facto de ser mais fácil e rápido de fazer a manutenção de um código único e devidamente organizado (Ribeiro, 2008).

d. frameworks

Uma *framework* pode ser entendida como uma estrutura real ou conceptual de suporte para resolução de um determinado problema. Esta geralmente envolve uma combinação de componentes e ferramentas para abstracção de código. No âmbito da

engenharia de software, uma *framework* é geralmente entendida como um modelo de trabalho, uma estrutura por camadas que indica que tipo de programas podem ser desenvolvidos e como se relacionam. Outra das vantagens inerentes ao uso de uma *framework* passa pela obrigação a toda a equipa envolvida a programar de forma consistente (Clifton, 2003). Existem dois tipos de *frameworks*, as de software que tipicamente compreendem um conjunto de classes numa determinada linguagem para orientação no desenvolvimento de software, e as *frameworks* conceituais que são um modelo de dados, um conjunto de conceitos passíveis de ser usados para a resolução de um problema especifico.

III. Projecto em contexto empresarial

1. Procedimento metodológico

Esta investigação seguiu uma metodologia mista, simultaneamente de estudo de caso e investigação-acção. A especificidade do ponto de vista operativo neste projecto aponta para um estudo de caso, uma vez que toda a investigação se centrou na realidade actual da empresa Dreamlab. Esta investigação teve ainda, claramente, uma forte componente de investigação-acção, requerendo um elevado nível de envolvimento participativo, que passou pela integração física no contexto de trabalho da empresa em questão, ainda que apenas temporariamente.

Como em qualquer outro projecto de investigação, a primeira fase passou pela correcta identificação do problema em questão. Este primeiro momento implicou um contacto físico com a empresa para uma mais clara definição e caracterização da problemática e definição dos passos a seguir para a sua resolução com sucesso, para ambas as partes envolvidas. Este primeiro contacto ocorreu ainda antes do inicio do projecto em contexto empresarial, para permitir uma recolha teórica dentro da área do desenvolvimento ágil para a Web o mais rica possível, parte também integrante destes estudo.

Já em contexto empresarial e após nova reunião ficou decidido que inicialmente haveria um período de integração e familiarização com o processo de desenvolvimento de conteúdos para a Web, na Dreamlab. Para tal, durante este período houve um acompanhamento presencial e observação directa do desenvolvimento de diversos projectos a decorrerem na altura, por parte do investigador. No final deste acompanhamento, foram realizadas entrevistas para complementar algumas fragilidades no acompanhamento e também ter uma visão mais abrangente do processo geral de desenvolvimento de conteúdos para a Web. Nesta etapa, os participantes do estudo foram os elementos da equipa de programadores, escolhidos pelo seu envolvimento mais prático na realização dos conteúdos e o orientador de projectos relacionados com a Web na empresa, para uma visão mais geral. Após esta recolha de dados foi realizada uma análise qualitativa da mesma, que será discutida mais adiante neste documento, para terminar o ciclo de integração e compreensão, mas também para servir de ponto de partida para a próxima fase, o desenvolvimento dos protótipos.

Após esta etapa ocorreu nova reunião com os orientadores da Dreamlab para discussão de quais os protótipos a desenvolver. Em conjunto, decidiu-se pelo

desenvolvimento de uma *newsletter* e de uma plataforma de partilha de ficheiros, esta última um projecto que estava em lista de espera para ser desenvolvida assim que o calendário da empresa assim o permitisse. Nesta altura foram definidos os resultados esperados para cada um dos protótipos, com mais ênfase nas especificidades do protótipo final. Foram decididos em conjunto também os requisitos funcionais para ambos os protótipos, sendo que o primeiro protótipo teria as especificações típicas de uma *newsletter* básica e o protótipo final iria consistir em dois grupos distintos de especificações: o primeiro grupo de especificações iria garantir o funcionamento básico da ferramenta e o segundo servia como exemplo de outros complementos passíveis de ser introduzidos na plataforma básica.

A partir desta altura, o investigador calendarizou o tempo para cada um dos protótipos, tendo sempre em conta uma primeira fase exploratória, uma fase seguinte para o desenvolvimento do protótipo e finalmente validação do mesmo. Após as fases de validação dos protótipos estavam sempre previstos alguns dias sem trabalho agendado, dedicados a eventuais alterações nos protótipos. Estes dias livres serviram também de segurança para eventuais atrasos imprevistos, de modo a não comprometer a viabilidade do estudo.

Durante todo o período de desenvolvimento dos protótipos a participação por parte de pessoal da empresa não foi significativa, com excepção do contacto em momentos pontuais com elementos do grupo de programação para esclarecimento de algumas dúvidas, cingindo-se normalmente apenas a receber um relatório semanal por parte do investigador, para controlo do decorrer do estudo realizado. Este cenário decorreu na natureza das tarefas em causa e não das dinâmicas de grupo da organização.

Os protótipos foram desenvolvidos inicialmente num servidor local a cargo do investigador por questões de comodidade; já na fase de validação final junto do *focus group*, foram colocados online num servidor público. As adaptações que esta mudança obrigou ao nível do código gerado nos protótipos, na sua maioria não foram significativas, prendendo-se essencialmente com alterações ao nível dos dados de acesso à base de dados. Como já foi referido, depois de desenvolvidos os dois protótipos, estes foram testados por elementos da própria equipa de desenvolvimento de conteúdos Web na Dreamlab, para um feedback mais técnico e valioso, dado por quem trabalha na área. Um resumo desse mesmo feedback conseguido pelo *focus group* encontra-se mais à frente neste documento.

2. Etapas do estudo

Terminado o período de familiarização com o processo de desenvolvimento de produtos Web na Dreamlab, seguiu-se a fase da análise qualitativa dos dados recolhidos anteriormente. Depois de uma reunião em conjunto com os orientadores começou a fase mais prática deste estudo: a prototipagem de módulos ágeis, escolhidos a partir de um consenso entre ambas as partes envolvidas nesta investigação. Desta reunião decidiu-se o desenvolvimento de um protótipo de *newsletter* e um protótipo de plataforma de partilha de ficheiros, assim como os principais requisitos para ambos os protótipos. Foi então definido um calendário que surgiu do cruzamento entre o tempo restante do investigador na empresa e a dificuldade aparente das especificações dos protótipos propostos.

Para o primeiro protótipo, o da *newsletter*, definiu-se um período de tempo de 10 dias úteis, mais 2 livres para segurança e revisões. Destes 10 dias, os primeiros 3 foram dedicados à exploração e desconstrução, não só de exemplos online, mas também de alguns exemplos de outras *newsletters* desenvolvidas anteriormente na Dreamlab e testes da autoria do próprio investigador. Seguiu-se a fase de prototipagem, durante um intervalo de tempo de 5 dias úteis, onde se desenvolveu o projecto piloto que foi testado, nesta fase, ainda que apenas pelo investigador. Em termos de funcionalidades, o que se esperava nesta fase era a o desenvolvimento de um protótipo capaz de receber contactos de email e guardá-los numa pilha, que viria posteriormente a servidor destino para um mass mail. Este mass mail teria ainda que permitir a inclusão de um anexo no formato JPG e os utilizadores teriam que ver disponível a opção de eliminarem o contacto deles em qualquer altura. Os 2 dias que se seguiram foram dedicados à validação do resultado final, agora sim por parte dos orientadores. Uma vez que os dias previstos como segurança não chegaram a ser usados, o projecto final começou mais cedo, numa tentativa de rentabilizar tempo.

Tendo em conta a aparente dificuldade acrescida da plataforma de partilha relativamente à *newsletter*, o calendário para desenvolvimento desta foi de 25 dias úteis. Uma vez mais, a primeira fase consistiu na exploração e desconstrução de exemplos, durante um período de 2 dias. Contrariamente ao que aconteceu no protótipo da *newsletter*, no caso da plataforma de partilha a empresa nunca tinha desenvolvido nada semelhante, estando previsto o desenvolvimento da mesma apenas quando o calendário da empresa assim o permitisse. A fase seguinte, a durar 21 dias úteis foi dividia em duas metades, tendo em conta os requisitos funcionais da plataforma. Numa primeira metade foram desenvolvidas as bases que garantissem o correcto funcionamento da plataforma,

bases estas que consistiam no upload de ficheiros, a inserção de um comentário associado ao mesmo e ainda as opções de listar e abrir ficheiros presentes no servidor remoto. Durante este período, tal como aconteceu anteriormente, todos os testes estavam a cargo apenas do investigador. Uma vez mais, concluída a fase de prototipagem seguiu-se a fase de validação onde os orientadores testam e avaliam os resultados obtidos. Depois desta fase, os protótipos foram considerados completos e enviados para os elementos integrantes da equipa de desenvolvimento de conteúdos Web da Dreamlab, para a realização de testes em *focus groups*. Depois de devidamente testados, os resultados foram considerados satisfatórios tendo o código fonte ficado disponível para subsequentes utilizações futuras por parte da empresa.

3. Modelo de análise

Dentro da temática em questão, facilmente se levantaram os dois conceitos centrais deste estudo - optimização e agilidade – cujas dimensões e indicadores se encontram apresentados na tabela 1 – Modelo de Análise.

O conceito de optimização remete para a análise de uma prática já implementada na empresa, de modo a que esta possa ser melhorada, em todas as componentes que a envolvem. Assim, dentro deste conceito realçaram-se as dimensões económica e recursos consumidos, estes últimos divididos em recursos humanos e temporais e também recursos tecnológicos. Os indicadores para a dimensão económica foram os custos e gastos provenientes no decorrer do processo de desenvolvimentos de conteúdos Web. Para as dimensões dos recursos, considerou-se pertinente dividir em duas categorias maiores, recursos humanos e temporais numa só, uma vez que os indicadores estão claramente interligados e são comuns a ambas e finalmente recursos tecnológicos, cujos indicadores foram tipicamente as ferramentas usadas. Quanto aos recursos humanos e temporais, os indicadores levantados foram o número de recursos dispendidos (número médio de pessoas envolvidas directamente no processo), o número de horas médio dispendidas por tarefa e finalmente o contributo para a formação pessoal de cada um dos envolvidos. Este ultimo indicador é relevante na medida em que é particularmente interessante dum ponto de vista de investigação saber se no final do processo de desenvolvimento de um serviço, a experiência foi enriquecedora ao nível do conhecimento, para a equipa envolvida.

O conceito da agilidade aponta para a facilidade, rapidez e eficiência de desenvolvimento; uma maior capacidade de resposta aos pedidos dos clientes e mais eficiente adaptação à mudança. É inserido neste conceito que surge o ponto fulcral desta investigação, sob a forma das dimensões reutilização e modularidade. Os indicadores da reutilização foram o tempo que se pode "ganhar" num processo e também o conhecimento novo que pode ser proveniente de uma abordagem mais ágil no desenvolvimento de um serviço. Dentro da modularidade os indicadores foram o número de módulos criados ou passíveis de ser criados assim como a diversidade dos mesmos, dentro de cada temática/utilização. Finalmente interessou também fazer um levantamento da elasticidade destes mesmos módulos, ou seja, a capacidade de um mesmo módulo poder ser adaptado a um outro tipo de serviços que não o inicialmente pensado, mediante pequenas e rápidas alterações.

Conceitos	Dimensões	Indicadores
Optimização	Económica	Custos
	Recursos humanos e temporais	Número de recursos
		Número de horas/tarefa
		Contributo para a formação
	Recursos tecnológicos	Ferramentas usadas
Agilidade	Reutilização	Tempo
		Conhecimento
	Modularidade	Número de módulos
		Diversidade de módulos
		Plasticidade dos módulos

Tabela 1 - Modelo de análise

4. Recolha de dados

Numa primeira fase, a recolha de dados foi realizada pelo método de observação directa, por parte do investigador, no período inicial de integração no grupo de trabalho da Dreamlab. Nesta etapa, foi acompanhado o desenvolvimento de diversos projectos que se encontravam em desenvolvimento para clientes reais, numa tentativa não só de compreensão da lógica de trabalho da empresa, mas também de levantamento de quais os pedidos mais recorrentes e serviços mais vezes implementados.

Posteriormente, a recolha de dados foi depois complementada através da realização de inquéritos por entrevista, inicialmente junto de elementos que constituíam a equipa de programadores e, seguidamente, junto do coordenador de projectos. Como já foi referido anteriormente neste documento, os objectivos das duas fases eram também

eles distintos, daí que tivessem sido realizadas junto de elementos diferentes e em diferentes calendarizações. Tendo como ponto de partida o modelo de análise anteriormente exposto, foram redigidos diferentes guiões para os inquéritos por entrevista. As perguntas efectuadas foram tipicamente de resposta fechada, com um número limitado de opções para a maioria das respostas e organizadas por grupo de temática. No final de cada grupo de questões, foi permitido aos entrevistados acrescentarem algo que considerassem relevante às respostas que já tinham dado anteriormente. As entrevistas foram realizadas verbalmente e, com a autorização de todos os entrevistados, foram registadas em formato áudio com o auxilio de uma ferramenta de gravação, para facilitar a posterior análise qualitativa das mesmas. Como já foi referido anteriormente, a fase de recolha dos dados e subsequente análise dos mesmos antecedeu o planeamento e desenvolvimento dos protótipos.

Para a entrevista a realizar junto dos elementos da equipa de desenvolvimento de conteúdos Web, foi realizado um guião que apontava para aspectos mais concretos e técnicos, formação de equipas e outros recursos consumidos. Este enfoque mais prático está relacionado com o primeiro conceito realçado no modelo de análise, o de optimização. Assim, esta primeira ronda de entrevistas foi realizada com o principal intuito de compreender o processo actual de desenvolvimento de conteúdos, para que após uma análise posterior, este pudesse eventualmente ser melhorado ou, pelo menos, complementado. Também contemplada neste mesmo guião surgiu já a questão da reutilização de serviços ou conteúdos anteriormente desenvolvidos.

A entrevista a realizar junto do coordenador de projectos teve como principal objectivo ter uma visão mais abrangente da realidade actual da Dreamlab face ao desenvolvimento de conteúdos Web. No entanto, interessou também saber qual a opinião das chefias face à agilidade de processos e que tipo de mentalidade tentam incutir às equipas que comandam. Esta segunda ronda de entrevistas aponta já para questões da agilidade, evidenciadas no modelo de análise, como é o caso da existência prévia de módulos já desenvolvidos e a plasticidade dos mesmos.

Os guiões para ambas as fases de entrevistas encontram-se na secção de anexos deste documento.

5. Análise preliminar das práticas da empresa

Ainda antes do desenvolvimento dos protótipos, foi analisado o processo de criação de conteúdos para a Web na Dreamlab. O objectivo desta análise foi, não só a familiarização com o processo em si, mas também perceber que tipo de preocupações a empresa tem neste momento, relativamente ao reaproveitamento de soluções e agilidade dos métodos utilizados. Durante a fase de acompanhamento foi perceptível que a empresa já tinha desenvolvidos módulos ágeis para montras de serviços/produtos, de *backoffice* e, ainda, de notícias, que são implementados regularmente. A criação destes módulos deveu-se também ao cliente típico da empresa e aos tipos de pedidos mais recorrentes, de modo a permitir agilizar o processo de criação de conteúdos para a Web.

Os módulos foram desenvolvidos de modo a serem plásticos o suficiente para poderem ser implementados em resposta a diversas situações, situação que está bem clara no módulo de montra que tanto serve para dar a conhecer produtos de uma empresa como mostrar os serviços disponíveis da mesma. Relativamente ao módulo de *backoffice* presente, este ainda continua a ser trabalhado actualmente com o objectivo de ser realmente modular. Durante a fase de discussão para decidir quais os protótipos a serem desenvolvidos ainda se falou em numa eventual optimização do módulo de *backoffice*, mas a ideia foi abandonada por este estar a ser usado em diversos projectos nesse momento, assim como a falta de familiarização do investigador relativamente aos processos já existentes poderia ser um factor negativo e levar a uma evolução temporal do trabalho mais lenta ou até mesmo negativa.

Relativamente ao módulo de notícias já criado, este permite a gestão automática de noticias por parte dos *webmasters* dos sites onde é inserido, e inclui ainda a possibilidade de destaques dentro do próprio conjunto de notícias existente. Esta preocupação pelo reaproveitamento de soluções parte dos gestores de projectos, mas faz também parte da mentalidade da própria equipa de programadores, uma vez que é prática corrente tentarem sempre ao máximo nunca começar um projecto totalmente do zero e aproveitar algo que já tenha sido desenvolvido anterior, mesmo que sejam necessárias modificações no novo contexto.

Ainda relativamente às práticas da empresa, constatou-se que, normalmente, quando um cliente efectua um novo pedido, a divisão de tarefas é feita tendo por base o nível de conhecimento dos elementos disponíveis, assim como a identificação destes face a pedidos semelhantes. Esta adaptação do trabalho aos indivíduos envolvidos tem base numa mentalidade de desenvolvimento ágil (Highsmith, 2004). Também a criação

de equipas está em conformidade com metodologias ágeis uma vez que, em condições normais são de dimensão reduzida, entre 3 a 4 elementos, com pelo menos um de cada área operacional necessária, geralmente design, animadores flash e programadores de bases de dados.

Outra preocupação da empresa que faz também parte de uma mentalidade de desenvolvimento ágil é o constante contacto e comunicação existente não só entre todos os elementos que compõe as equipas, mas também com os orientadores de projecto e os próprios clientes. Este contacto constante entre equipas e clientes baseia-se num modelo de colaboração e constante interacção com vista no desenvolvimento do melhor produto final possível, acompanhado toda a fase de desenvolvimento.

Na Dreamlab foi ainda visível a preocupação pela criação de software funcional constantemente, para que os clientes possam acompanhar o evoluir dos pedidos e sempre que considerarem necessário, dar *input* valioso às equipas de desenvolvimento. Também este ênfase em software funcional em detrimento de documentação vem descrito no manifesto ágil e é mais uma indicação da mentalidade nitidamente ágil que é incentivada por parte dos gestores na empresa.

Pelo que foi possível observar, o modelo de negócio na parte do desenvolvimento para a Web, na Dreamlab, baseia-se muito numa lógica de comunicação e interacção entre a empresa e os clientes, com os últimos sempre a acompanhar o desenvolvimento dos seus pedidos e a dar feedback a quem, de facto, os está a desenvolver.

Esta constante preocupação e procura por uma permanente agilidade de processos está ainda presente nos módulos que foram escolhidos para protótipos desta investigação, principalmente a plataforma de partilha de ficheiros, cujo principal objectivo é o de facilitar e maximizar a interacção entre a empresa e clientes.

6. *Proposta e desenvolvimento dos módulos*

a. Módulos escolhidos

Com base nos dados apurados nas entrevistas realizadas pôde-se concluir que o cliente tipo da Dreamlab pertencerá, na sua maioria, ao contexto empresarial. Por norma este tipo de clientes possui um catálogo de produtos ou serviços que quer mostrar ao seu público-alvo. Para este efeito, a Dreamlab já tem devidamente desenvolvido um módulo plástico o suficiente, que tanto serve para produtos como para serviços, que é implementado regularmente. No entanto pode-se considerar pertinente que estas empresas não só disponibilizem catálogos de produtos, mas que disponham também da possibilidade de contactar directamente os seus clientes interessados neste efeito. Este tipo de contacto é especialmente eficaz para a promoção de produtos com base em compras anterior ou simplesmente manter os clientes interessados a par de eventuais promoções e novidades. Para isso o projecto piloto proposto para desenvolvimento foi o de uma *newsletter*. Esta *newsletter* consiste num módulo que quando aplicado em qualquer site, permite aos utilizadores inserirem o seu contacto de e-mail, para assim se demonstrarem disponíveis e interessados em receber mais informação da empresa no futuro, de forma automática através desse mesmo contacto.

Outra funcionalidade cuja implementação pode facilitar a comunicação e, consequentemente agilizar o desenvolvimento dos produtos por parte da empresa, passa pela criação de uma plataforma de partilha de ficheiros. Pelo que foi possível reter da análise inicial feita com o objectivo de compreender a lógica de funcionamento da Dreamlab, algo que acontece várias vezes ao longo do ciclo de desenvolvimento de um produto é o envio de propostas e protótipos para o cliente para este validar. Normalmente esta troca ocorre através do envio de e-mails, aos quais o cliente responde, por vezes também com outros ficheiros anexados, ou revisões dos mesmos. A implementação desta plataforma de partilha vai assim permitir que, após uma autentificação dos utilizadores, a troca de recursos entre clientes e empresa seja facilitada, através do envio de conteúdos para um espaço físico disponível para os clientes e empresa em simultâneo. Esta plataforma foi ainda acrescida das funcionalidades de listagem dos conteúdos presentes e informação relativa aos mesmos, como quem fez o upload, quando e um eventual comentário associado ao ficheiro. Este último campo tem também como objecto maximizar o contacto entre as duas entidades, permitindo passar feedback de um lado para o outro.

Tendo em conta a relativa facilidade de implementação da *newsletter* em relação à plataforma de partilha de ficheiros, a primeira foi desenvolvida sob a forma de projecto piloto, deixando assim mais tempo livre para o restante módulo. Também o facto de ser aparentemente mais simples permitiu uma maior fase de exploração no caso da *newsletter* intencionalmente, principalmente em rotinas e lógicas que já se previa que viriam mais tarde a ser reaproveitadas na plataforma de partilha de ficheiros. É ainda de referir que, em ambos os casos, não houve qualquer tipo de preocupação ao nível do design, uma vez que o ponto fulcral deste estudo é o reaproveitamento de funcionalidades. Com as rotinas de código correctamente desenvolvidas, estas estão prontas a ser aplicadas em plataformas como o Flash ou outras, para integração nos pedidos dos clientes, sempre que assim for necessário, sem que haja qualquer tipo de choque ou conflitos de design com restantes elementos presentes.

b. Protótipo piloto: newsletter

i. Requisitos funcionais

Como já foi referido anteriormente, para piloto foi escolhido o desenvolvimento de uma *newsletter*, parte fundamental da comunicação entre a empresa e clientes. Para o desenvolvimento do projecto piloto foram definidos como requisitos funcionais apenas as funcionalidades base para o correcto funcionamento de uma *newsletter* típica. Assim, identificaram-se como requisitos:

- criar subscrição;
- cancelar subscrição pelo *backoffice*;
- cancelar a própria subscrição;
- enviar um *mass mail* para todos os contactos;
- anexar um ficheiro jpg ao *mass mail*.

ii. Desenvolvimento

O desenvolvimento deste protótipo foi conseguido maioritariamente através da implementação da tecnologia server-*side* php, que permite a criação de páginas Web dinâmicas. Houve ainda a ligação a uma base de dados específica para armazenar e gerir todos os contactos. Para efeitos de teste foi criada uma página genérica em html com diversos formulários para poder aceder a todas as funcionalidades a implementar para a *newsletter*. Assim, nesta página estão presentes 3 formulários, um para adicionar uma subscrição, outro para cancelar a subscrição e finalmente um para inserir um assunto, escolher uma imagem e enviar. Houve ainda o acrescento de um link simples para uma listagem completa de todos os contactos presentes na base de dados.

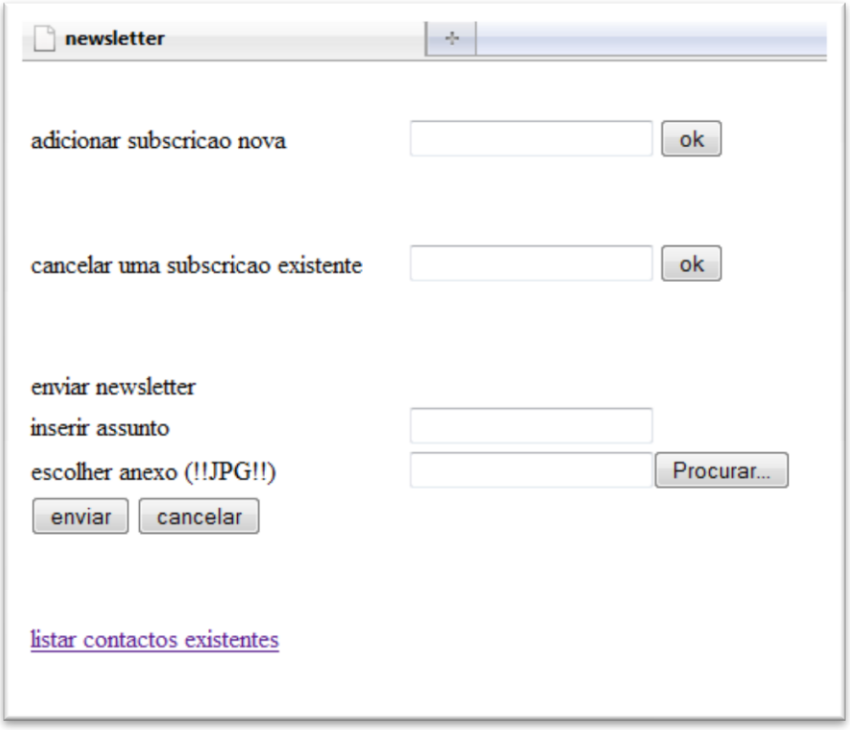

Ilustração 2 - Protótipo de newsletter

Uma vez que a *newsletter* funciona à base de contactos de email, por questões de segurança o método usado em todos os formulários foi o *post*, ou seja, qualquer parâmetro passado não é visível na barra de endereços do browser, e aponta para uma página distinta em php.

```
<form method=’post’ action=’’>
```

O desenvolvimento da *newsletter* começou pela funcionalidade de criar subscrição. Para tal foi necessária a criação de uma base de dados em mysql onde fossem inseridos os contactos que recebidos do formulário anterior. Através da função *mysql_query()* foi possível assegurar que o php executa as instruções mysql necessárias. Os contactos são guardados numa variável “$contacto” para serem posteriormente inseridos na base de dados, mas não antes de serem testados pela sua validade. Este teste é efectuado pela função *strisrt()* e incide em 2 parâmetros distintos, a existência de uma arroba (@) e de um ponto (.) no texto inserido. Esta medida, não sendo completamente à prova de erro, previne minimamente a inserção de contactos incorrectos. Apenas se o contacto inserido for verdadeiro nos 2 parâmetros é que este é inserido, senão é mostrada uma mensagem de erro ao utilizador.

```
$contacto = $_POST["contacto"];
if(!stristr($contacto,"@") OR !stristr($contacto,"."))
{
   echo “email invalido”
}
```

Caso o email seja válido é efectuada uma ligação à base de dados com a função *mysql_connect()* e, caso esta seja realizada com sucesso é tentada a criação de uma tabela dedicada nova. Esta criação serve como segurança para garantir que caso a tabela não exista, ela é criada, situação que acontece, por exemplo, quando é inserido o primeiro contacto, não obrigado assim à criação de uma tabela anteriormente por *backoffice*. Uma vez que já foi efectuado anteriormente um teste à criação da base de dados, neste momento parte-se do principio que o utilizador tem privilégios de criação e escrita, senão este ou qualquer outro erro que tenha ocorrido é mostrado ao utilizador logo de início através da função *mysql_error()* e não tem como seguir em frente. No protótipo criado a base de dados chama-se “basedados” e a tabela onde são inseridos os contactos tem o nome de “listagem”. A tabela criada tem apenas um campo textual obrigatório com, no máximo, 25 caracteres.

Este protótipo foi inicialmente desenvolvido e testado localmente e, portanto, os dados da ligação de acesso à base de dados são “localhost” para o servidor, “root” para o nome do utilizador e não foi usada nenhuma password.

```
$ligacao = mysql_connect("localhost","root");
if (!$ligacao)
{
  die ("nao foi possivel efectuar a ligacao" . mysql_error());}
  if (mysql_query("CREATE DATABASE basedados",$ligacao))
  {
    $tabela = "CREATE TABLE ‘basedados’.’listagem’ (‘contacto’ VARCHAR(25)
NOT NULL) ENGINE = MyISAM";
    mysql_query($tabela);
```

Após efectuada a ligação à base de dados e assegurado o correcto destino da informação, pode-se então passar à introdução do contacto na base de dados. Para isso criou-se, uma vez mais, uma *query* em mysql que indica qual o valor a guardar e onde. Finalmente, e após concluída com sucesso a inserção do contacto, falta apenas terminar a ligação com a base de dados para ficar completo o primeiro requisito funcional.

```
mysql_query("INSERT INTO ‘basedados’.’listagem’ (‘contacto’) VALUES (‘".
$contacto ."’)");
mysql_close(‘ligacao’);
```

Para cancelar uma subscrição pelo *backoffice*, funcionalidade esta que ficará disponível apenas para os gestores do site, usa-se o método *post* para passar o contacto do formulário para uma página php para o efeito, onde este vai ser guardado numa variável "$cancelar". É ainda criada uma nova variável chamada $existe, que vai servir como *flag*, apenas para guardar o estado de existência de um contacto correspondente e assume o valor 0 inicialmente.

```
$existe = 0;
$cancelar = $_POST["contacto"];
```

Depois de testada a ligação à base de dados, com uma lógica em tudo semelhante à anteriormente usada, é seleccionada a base de dados e efectuada uma pesquisa no campo "contacto" da tabela "listagem" por um valor igual ao introduzido no formulário. Para este efeito é usada a função *mysql_fetch_assoc()*, que retorna um conjunto de *strings* se encontradas ou então o valor *false*, caso contrário. Quando encontrados contactos iguais aos inseridos no formulário de raiz, estes são apagados da base de dados. Caso não ocorra nenhum resultado da pesquisa, o utilizador é informado de que o contacto inserido não se encontra na base de dados.

```
mysql_select_db(basedados, $ligacao);
$query = mysql_query("SELECT *  FROM 'listagem' WHERE 'contacto' LIKE '" .
$cancelar . "'");
while ($row = mysql_fetch_assoc($query))
{
  $existe = 1;
}
if ($existe==1)
{
  mysql_query("DELETE FROM 'basedados'.'listagem' WHERE 'listagem'.'contacto'
= '" . $cancelar . "'");
  echo "o contacto " . $cancelar . " foi retirado da base de dados com
sucesso ";
}
else
{
  echo "contacto " . $cancelar . " nao encontrado";
}
```

Terminada a pesquisa e eventual eliminação de registos, a ligação com a base de dados é desligada, tal como acontecia na funcionalidade anterior. Segue-se a funcionalidade principal deste módulo, o envio de um *mass mail* para todos os contactos que estão na base de dados. Aqui a maior dificuldade e onde pode por em causa o funcionamento do email em si, é na correcta definição dos *headers*. Para envio de um

email simples, basta usar a função *mail()*, mas para enviar uma mensagem com anexos é necessário criar mensagens do tipo MIME[4], correctamente definidas e construídas. Neste formato além de ser necessário indicar o destino, o assunto e o corpo da mensagem, é ainda obrigatório o envio do *header*, depois de devidamente definido. Para isso é necessário obedecer a uma estrutura fixa de instruções, tanto nos *headers* onde é indicada a versão de MIME a usar, o tipo de conteúdo da mensagem (texto simples, texto com formatação html ou anexos) e os limitadores (*bounds*), como no próprio corpo da mensagem, no qual é necessário indicar o tipo de conteúdo e a codificação para as diferentes partes da mensagem, sempre divididas pelo limitador definido no *header*. Para proceder ao envio de um email com anexo é preciso, uma vez mais, receber a informação que vem do formulário anterior através do método *http post* onde, além do assunto, é ainda necessário guardar o nome do ficheiro anexo, assim como a sua localização temporária no servidor. Estes valores devem ser guardados em separado uma vez que um faz referência apenas ao nome, enquanto que o outro aponta para o conteúdo real do ficheiro. Por questões de agilidade de código pode-se ainda definir nesta fase o limites do corpo da mensagem.

```
$assunto = $_POST["assunto"];
$ficheiro = $_FILES["ficheiro"]["name"];
$anexo = $_FILES["ficheiro"]["tmp_name"];

$bound_text = "boundry generico"; //para alterar
$bound = "--". $bound_text. "\r\n";
$bound_last = "--". $bound_text. "--\r\n";
```

De seguida, testa-se, pela existência de um anexo e do assunto da mensagem, uma vez que são ambos campos obrigatórios para o envio do email e, caso estejam presentes, tenta-se a ligação à base de dados, tal como acontecia nas funcionalidades anteriores. No caso de estar tudo em conformidade, faz-se uma listagem de todos o conteúdo na tabela “listagem” da base de dados e segue-se para a codificação do anexo. Começa-se então por abrir o ficheiro para leitura em modo binário com a função *fopen()*, seguida de uma leitura completa de todo o ficheiro, através da função *fread()*. Depois de aberto e lido na sua totalidade, o ficheiro tem que ser dividido em partes, através da função *chunk_split()*, que divide uma *string* longa em partes iguais mais curtas e codificam-se essas partes em *base64*, que permite a que a informação sobreviva ao transporte através do corpo do email. Após a informação do anexo ser devidamente codificada e guardada numa variável, o acesso ao ficheiro pode ser fechado, com a

[4] MIME – multipurpose internet mail extensions

função *fclose()*. Nesta altura pode-se já definir também os *headers* do email a enviar posteriormente, sendo neste caso uma mensagem mista com várias partes.

```
$abre = fopen($anexo, "rb");
$anexo = fread($abre, filesize($anexo));
$anexo = chunk_split(base64_encode($anexo));
fclose($abre);

$headers = "From: <remetente>\n";
$headers .= "MIME-Version: 1.0\n";
$headers .= "Content-Type: multipart/mixed; boundary=\"$bound_text\"";
```

De seguida, cria-se um ciclo que vai se vai repetir para todos os registos presentes na base de dados e que vai criar o corpo da mensagem a enviar. Uma vez que as *newsletters,* tipicamente, disponibilizam um link para cancelar a subscrição na própria mensagem, o corpo da mensagem a enviar vai ser composto por duas partes distintas. Uma primeira parte composta por texto em html, onde o php vai passar através do link a identificação do utilizador que recebeu a mensagem, para que possa cancelar a subscrição sem precisar de introduzir nada, e uma segunda parte com o anexo em si, já devidamente codificado no passo anterior e pronto a enviar/receber. No caso deste protótipo, a página "cancelar2.php" é onde se vai processar o cancelamento da subscrição, via o link enviado no email. A página de destino usada foi outra que não a anteriormente criada para cancelar subscrições por um detalhe que irá ser descrito e explicado mais à frente. O fecho do corpo da mensagem deve ser delimitado com o último *bound*, diferente dos *bounds* intermédios. Neste momento o email está devidamente formatado e pronto a ser enviado, um de cada vez para cada contacto na lista, com a função *mail()*, onde o destino será o resultado de cada pesquisa efectuada à base de dados, o assunto é o que foi inserido no formulário inicial e o corpo da mensagem e *headers* foram definidos anteriormente. Após o envio dos emails, pode-se dar algum tipo de feedback à empresa sobre quais seguiram com sucesso e após este processo, a ligação à base de dados já pode ser terminada.

```
while ($row = mysql_fetch_assoc($query))
{
  $mensagem .= "Content-Type: text/html; charset=\"iso-8859-1\"\r\n";
  $mensagem .= "Content-Transfer-Encoding: 7bit\r\n\r\n";
  $mensagem .= "<a href=cancelar2.php?contacto=". $row[‘contacto’] .”> clique
aqui para cancelar a subscricao </a>";
  $mensagem .= $bound;

  $mensagem .= "Content-Type: image/jpg; name=" . $ficheiro ." \r\n";
  $mensagem .= "Content-Transfer-Encoding: base64 \r\n";
  $mensagem .= "Content-disposition: attachment; file=" . $ficheiro ." \r\n";
  $mensagem .= $anexo;
  $mensagem .= $bound_last;

  if (mail($row[‘contacto’], $assunto, $mensagem, $headers))
  {
    echo "newsletter enviada com sucesso para " . $row[‘contacto’]";
  }
}
mysql_close(‘$ligacao’)
```

Como foi referido anteriormente, em cada email enviado é inserido um link para que o destinatário possa cancelar a sua subscrição. O processo para cancelar a subscrição já foi descrito e a única diferença é que, uma vez que a informação desta vez é passada pelo link, o método usado para a recuperar terá que ser o *get*, em vez do *post* usado anteriormente para todos os casos. Exceptuando esta pequena alteração, toda a lógica anterior é mantida e inalterada para o resto do código criado.

```
$mensagem .= "<a href=cancelar2.php?contacto=". $row[‘contacto’] .”> clique
aqui para cancelar a subscricao </a>";

$cancelar = $_GET["contacto"];
```

Finalmente, na página inicial além de todos os formulários necessários para aceder às funcionalidades desenvolvidas, foi ainda acrescentada a opção de listar e contar todos os contactos presentes na base de dados a qualquer dado momento. Para tal, desta vez não é necessário passar valores nenhuns entre diferentes páginas, basta apenas criar de novo uma ligação à base de dados, fazer uma *query* em sql para seleccionar todos os campos presentes e mostrá-los um a um. Para efeitos de contagem pode-se criar uma variável com o valor inicial de 0 e ir incrementando-a para que no final seja possível saber imediatamente quantos contactos estão na base de dados, ser que seja assim preciso estar a contá-los manualmente.

```
$contactos = 0;

$ligacao = mysql_connect("localhost","root");
if (!$ligacao)
{
  die ("nao foi possivel efectuar a ligacao <br>" . mysql_error() . "<br>");
}

mysql_select_db(basedados, $ligacao);
$query = mysql_query("SELECT * FROM 'listagem'");
while ($row = mysql_fetch_assoc($query))
{
  echo $row['contacto'] . "<br>";
  $contactos++;
}

echo "<br> de momento existem " . $contactos . " contactos validos";
```

Falta apenas referir que, uma vez que foram desenvolvidas várias páginas diferentes, com a excepção da página com os formulários (a página inicial, "newsletter.html"), em todas as restantes foi ainda usada a função *header()* do php para criar um cabeçalho de página que vai garantir que passado um determinado intervalo de tempo, o *browser* carregue a página inicial. Este tipo de solução acaba com a necessidade de usar o botão de retrocesso dos *browsers* que iriam enviar de novo informação para o servidor, ou então a obrigatoriedade de fecho de uma página para voltar às anteriores. Esta função tem que ser usada antes de haver qualquer tipo de *output* em todas as páginas. A inclusão desta nova linha de código deve-se a que todas as restantes páginas desenvolvidas são apenas de processamento, não havendo assim justificação para que se mantenham abertas indefinidamente. O tempo escolhido foi de 5 segundos, que permite atentar calmamente a qualquer *feedback* proveniente do processamento sem ser demasiado lento na transição, mas pode ser alterado no código.

```
header("refresh:5; url=newsletter.htm");
```

c. Protótipo final: plataforma de partilha

i. Requisitos funcionais

Para o protótipo final foram definidos dois níveis de requisitos funcionais distintos, sendo que o primeiro nível garantia o funcionamento básico da plataforma e o segundo iria acrescê-la de outras funcionalidades. Assim, foram definidos como requisitos funcionais primários:

- fazer o upload de ficheiros para um servidor remoto;
- associar um comentário livre ao ficheiro;
- listar todos os ficheiros presentes no servidor;
- apagar um ficheiro presente no servidor;
- abrir todos os ficheiros presentes no servidor.

Com estes requisitos assegurados e a funcionar correctamente, a plataforma está pronta para a implementação de novas funcionalidades, que se revelem como mais valias para o serviço prestado, das quais se realçaram como requisitos funcionais secundários:

- autenticação de utilizadores;
- diferentes privilégios para os utilizadores;
- limitar o upload de ficheiros.

ii. Desenvolvimento

Também este protótipo foi desenvolvido com recurso às tecnologias dinâmicas php e mysql. Para o upload de ficheiros foi criado um formulário com 2 campos de *input* distintos, para permitir aos utilizadores procurarem localmente um ficheiro para fazer o upload e uma área de texto para associarem um comentário.

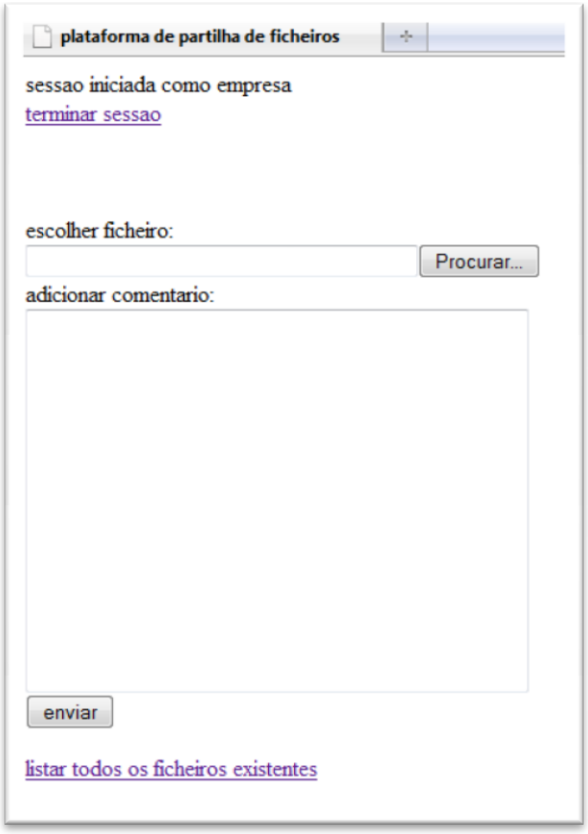

Ilustração 3 - Protótipo da plataforma de partilha

Uma vez mais, o método usado foi o *post*, para que não seja visível qualquer tipo de informação nos links. Inicialmente a página, apelidada de "main" foi criada em html, mas posteriormente foi alterada para uma página php, de modo a executar as funções dinâmicas que foram inseridas posteriormente. O destino do formulário é onde se vai efectivamente processar o upload dos ficheiros, na página "upload.php". Depois do formulário, foi criado um link simples para uma página "listar.php", onde vão ser listados todos os ficheiros presentes no servidor remoto, assim como toda a informação para cada um deles.

```
<form action="upload.php" enctype="multipart/form-data" method="post">
  <br> escolher ficheiro: <br>
  <input type="file" name="ficheiro" size="40">
  <br> adicionar comentario: <br>
  <textarea name="comentario" rows="15" cols="40"></textarea>
  <br>
  <input type="submit" value="enviar">
</form>
```

Tal como acontecia no protótipo da *newsletter*, também a plataforma de partilha precisa de comunicação com uma base de dados, para armazenamento e gestão da

informação enviada para o servidor, para garantir que esta não é perdida no final de cada sessão. Assim, na página de processamento do upload é necessário, antes de mais, garantir o acesso à base de dados e correcta criação das tabelas e campos necessários. A lógica é a mesma que a usada anteriormente, com uma diferença: a ligação à base de dados só vai ser tentada caso tenha sido escolhido um ficheiro para upload. Para tal é necessário criar variáveis para armazenar o ficheiro e o comentário e, ainda antes de proceder à ligação à base de dados testar pela validade do ficheiro, nomeadamente, se existe ou não.

```
$ficheiro = $_FILES["ficheiro"];

if ($ficheiro["error"] > 0)
{
  echo "nenhum ficheiro seleccionado";
}
else
{
  $ligacao = mysql_connect("localhost","root");
```

Depois de testada e efectuada a ligação, é criada a base de dados “basedados” e a tabela “upload”. A tabela criada tem que garantir a informação relativa ao ficheiro, como é o caso do nome do mesmo, o comentário associado e a data de upload, logo, a *query* em sql para a criação da tabela tem que visar todos os campos, nos seus tipos correctos. Para o efeito, uma vez mais é usada a função php *mysql_query()*.

```
mysql_query("CREATE TABLE ‘basedados’.’upload’ (‘nome’ VARCHAR(25) NOT NULL,
‘coment’ TEXT NOT NULL, ‘data’ TEXT NOT NULL) ENGINE = MyISAM;");
```

Para o php retornar dinamicamente a data corrente, usou-se a função *date()* para o efeito, inserindo como parâmetros do formato da mesma os caracteres “F” para representação textual completa do mês, “j” para dias do mês sem o zero antecedente, Y para representação completa do ano com 4 dígitos, e finalmente G:i para formato de 24 horas com zeros antecedentes. O formato final da data é guardado na variável “$data”.

```
$data = date("F j, Y, G:i");
```

Com a tabela criada com sucesso e ainda antes de guardar o ficheiro no servidor, é necessário testar para ver ser não há no servidor já outro com o mesmo nome, evitando assim erros ou sobreposição indesejada de ficheiros. Para este efeito o php tem a função *file_exists()* que vai retornar um valor booleano (verdadeiro ou falso) caso um ficheiro ou directoria exista ou não. No uso desta função é necessário ter em atenção os caminhos

dos ficheiros completos, no caso deste protótipo, os ficheiros são guardados remotamente numa directoria chamada de “files”

```
if (file_exists("files/" . $ficheiro["name"]))
{
  echo "o ficheiro ja existe";
}
else
{
```

Caso o ficheiro ainda não exista no servidor, está então tudo pronto para ser efectivamente feito o upload. Para isso usa-se a função *move_uploaded_file()* do php para mover o ficheiro da sua posição temporária de onde desapareceria no final da sessão para o espaço físico final dedicado ao efeito. De seguida insere-se nos campos correspondentes da tabela na base de dados a restante informação do ficheiro, o seu nome, o comentário e a data do upload, sendo que o nome e o comentário são provenientes do formulário da página inicial e a data provém dinamicamente da função anteriormente explicada.

```
move_uploaded_file($ficheiro["tmp_name"], "files/". $ficheiro["name"]);
mysql_query("INSERT INTO ‘basedados’.’upload’ (‘nome’, ‘coment’, ‘data’)
VALUES (‘". $ficheiro["name"] ."’, ‘". $comentario ."’, ‘". $data ."’)");
```

Outro requisito funcional principal que falta abordar é a opção para listar todos os ficheiros presentes no servidor. Aqui a lógica foi, uma vez mais, semelhante à já usada no protótipo da *newsletter*, sendo que inicialmente é testada a ligação à base de dados e, se esta for efectuada com sucesso, é realizada uma *query* em sql para retornar todos os valores presentes na tabela. Ainda que anteriormente tenha sido referido que nos protótipos não existiam preocupações de design, para uma melhor organização da informação retirada da base de dados, neste caso foram usadas tabelas simples de html, onde cada campo iria ocupar uma célula da tabela gerada. Esta tabela é gerada dinamicamente, consoante o numero de ficheiros disponíveis na base de dados, aumentando sempre que necessário. Para tal, criou-se primeiro apenas o cabeçalho da tabela e, depois de realizada a *query*, foi criado um ciclo que para cada ficheiro encontrado, é gerada uma linha completa nova e a tabela só é fechada já depois de terminado o ciclo. É a partir desta listagem que se pode abrir ou fazer o download dos ficheiros presentes no servidor, para isso foi criado um link directo para o ficheiro na célula com o nome do mesmo. Os ficheiros são abertos num separador novo (target =

_blank) para que não ocorra qualquer interferência com a consulta do estado do servidor. Para que os ficheiros possam ser apagados do servidor, e a sua informação removida da base de dados, é necessário que em cada uma das linhas criadas seja inserida uma célula com a opção de apagar o ficheiro. Para isso basta criar um link normal para a página de apagar ficheiros (no caso do protótipo "apagar.php") e passar como parâmetro o nome do ficheiro em causa, que surge dinamicamente através da consulta em sql.

```
mysql_select_db(basedados, $ligacao);
$query = mysql_query("SELECT * FROM 'upload'");
echo "<div align=right><br> <a href='main.php'> voltar </a></div>";
echo "<table border=10 width=100%>";
echo "<tr> <td align='center'> <b> nome </b> </td> <td align='center'> <b>
apagar </b> </td> <td align='center'> <b> comentario ao upload </b> </td> <td
align='center'> <b> upload feito em </b> </td> </tr>";

while ($row = mysql_fetch_assoc($query))
{
  echo "<tr><td> <a href='files/" . $row['nome'] . "' target='_blank'> " .
$row['nome'] . "</a> </td> <td align='center'> <a href='apagar.php?ficheiro="
. $row['nome'] . "'> <b> apagar <b> </a> </td> <td> " . $row['uploader'] . "
</td> <td> " . $row['coment'] . " </td> <td> " . $row['data'] . " </td>";
}

echo "</table>";
```

Tal como acontecia anteriormente, para se poder apagar um registo da base de dados, neste caso a informação referente a um ficheiro, é primeiro necessário receber o valor que é passado através do link referente ao nome do ficheiro, pelo protocolo "get". Depois de guardado numa variável este valor, usa-se a função php *unlink()* para apagar fisicamente o ficheiro do servidor e, caso o ficheiro tenha sido removido com sucesso, faz-se uma nova consulta sql à base de dados pelo mesmo ficheiro e apaga-se a entrada completa, com toda a sua informação. O procedimento aqui foi em tudo semelhante ao usado anteriormente no protótipo da *newsletter*, com excepção apenas no uso da função *unlink()* para remover o ficheiro do servidor. Caso o ficheiro não seja removido com sucesso, a base de dados permanece inalterada e o utilizador é informado de que ocorreu um erro durante o processo.

```
$ficheiro = $_GET["ficheiro"];
if (unlink("files/" . $ficheiro))
{
  mysql_select_db(basedados, $ligacao);
  $query = mysql_query("SELECT * FROM ‘upload’ WHERE ‘nome’ LIKE ‘" .
$ficheiro . "’");
  while ($row = mysql_fetch_assoc($query))
  {
    mysql_query("DELETE FROM ‘basedados’.’upload’ WHERE ‘upload’.’nome’ = ‘"
. $ficheiro. "’");
    echo "o ficheiro " . $ficheiro . " foi apagado com sucesso <br>";
  }
}
else
{
  echo "erro ao apagar o ficheiro";
}
```

Com a opção para apagar ficheiros remotos concluída, os requisitos funcionais principais ficaram todos cumpridos e, após uma curta fase de validação, começaram a ser desenvolvidos e postos em prática os requisitos funcionais secundários, que viriam a complementar a plataforma. O primeiro, e sem duvida mais importante, foi o desenvolvimento de um sistema de autenticação de utilizadores, já que a plataforma se destina à comunicação entre empresa e clientes e não deve portanto ter acesso aberto ao público em geral. Além de restringir o acesso à plataforma, a inclusão do sistema de autenticação visava ainda a diferenciação entre utilizadores, em relação aos privilégios que cada um possui. Para a autenticação foi criada uma nova página php, na qual foi usada a função *session_start()*, que começa ou resume uma sessão iniciada e criou-se uma variável “$erro” inicialmente vazia, para dar feedback ao utilizador caso os logins inseridos estejam errados.

```
session_start();
$erro = ‘’;
```

De seguida criou-se, seguindo a estrutura de uma página html típica, (entre as tags <html> e </html>) um formulário com dois inputs de texto, um para o login e outro para a password, cujo destino é a própria página onde se encontra, “login.php”, uma vez que o código de processamento do login viria a ser inserido na mesma página posteriormente. Imediatamente antes do formulário, foi inserida em php a variável criada para que o utilizador tenha feedback, em caso de erro.

```
<html>
  <head><title>login</title></head>
  <body>
    <?php
      if ($erro != '') {
        echo “<b> . $erro . “</b>”;
      }
    ?>
    <form method="post" name="login">
```

O formulário criado é apenas o output visual, não tendo ainda qualquer tipo de processamento. Para isso criou-se em php, antes da estrutura da página html, uma rotina para testar a validade dos valores inseridos nos inputs do formulário. Através de uma estrutura de decisão simples, inicialmente os dois inputs são testados para ver se estão preenchidos ou não, e só no caso de estarem ambos preenchidos, é que são posteriormente confirmados os valores inseridos. Para o protótipo foram criados dois logins diferentes, “empresa” e “cliente”, ambos com a password “teste”, logins estes cujo valor fica guardado numa variável de sessão chamada “logado”. É ainda de referir que nunca está disponível a opção de criar uma conta nova, em qualquer altura do funcionamento da plataforma. Esta opção foi deliberada, por se ter entendido que uma vez que se trata de uma plataforma de uso restrito, não tinha lógica que qualquer utilizador pudesse criar uma conta à vontade, só por ter acesso ao link da mesma. Se, de facto, os dados inseridos no formulário coincidirem com os valores rígidos presentes no código, então o utilizador é reencaminhado para a página “main.php”, e é guardada numa variável de sessão qual dos logins foi usado. Caso os dados inseridos não sejam correctos, a variável com a mensagem de erro é alterada, ficando assim visível ao fazer o *reload* da página.

```
if (isset($_POST[‘input_login’]) && isset($_POST[‘input_pass’])) {
  if ($_POST[‘input_login’] == ‘empresa’ && $_POST[‘input_pass’] == ‘teste’){
    $_SESSION[’logado’] = "empresa";
    header(‘location: main.php’);
    exit;
  } elseif ($_POST[‘input_login’] == ‘cliente’ && $_POST[‘input_pass’] ==
‘teste’) {
    $_SESSION[‘logado’] = "cliente";
    header('location: main.php');
    exit;
  } else {
    $erro = “dados errados, tente novamente”;
```

Neste momento, a autentificação de utilizadores está a funcionar, no entanto, para efectivamente restringir o acesso à plataforma seria preciso que os utilizadores acedessem à mesma manualmente através da página de login, e mesmo assim, depois de acederem não haveria ainda qualquer tipo de feedback ou uso real da autenticação. Para isso, a página inicial, “main.php”, foi alterada. Antes da estrutura html, foi incluída uma rotina de php para que, depois de retomada a sessão com a função *session_start()*, seja testada a variável de sessão criada anteriormente para verificar se esta assumiu o valor relativo a algum dos logins existentes. Em caso negativo, o utilizador é reencaminhado automaticamente para a página de login. Com a inclusão deste teste, a página de acesso volta a poder ser a “main.php”, uma vez que ainda antes de haver qualquer tipo de output para o *browser*, é efectuado um teste ao login usado e se usado. Exactamente a mesma lógica foi usada e introduzida na página “listar.php”, obrigando assim a autentificação do utilizador para visualização desta página. A partir deste momento, ambas as páginas reencaminham o utilizador para a página de login, caso a autentificação não tenha sido efectuada, mesmo que o utilizador escreva no browser o endereço directo de qualquer uma delas. Esta obrigação funciona também como um reforço de segurança e privacidade da informação presente na plataforma.

```
session_start();

if (!isset($_SESSION['logado']) || $_SESSION['logado'] == "") {
  header('location: login.php');
  exit;
}
```

Foi ainda criada uma rotina de código para dar ao utilizador feedback sobre qual a sessão iniciada. Para o efeito, ainda na página “main.php”, foi inserida antes do formulário para o upload dos ficheiros, uma estrutura de decisão simples que retorna os valores “empresa” ou “cliente”, consoante a sessão iniciada anteriormente.

```
sessao iniciada como
<?php
if ($_SESSION['logado'] == "empresa") {
  echo  "empresa";
} elseif ($_SESSION['logado'] == "cliente") {
  echo  "cliente";
}
?>
```

Com a introdução do sistema de diferenciação de utilizadores, a informação enviada para o servidor durante os uploads também foi revista. Além do nome do ficheiro, comentário e data, deve também ser enviado para o servidor o nome do utilizador que fez o upload, para uma mais correcta identificação e controlo do uso da plataforma. Para tal, teve quer introduzido um novo campo na base de dados, denominado de "uploader" para guardar o login do utilizador e também a expressão para dar entrada de toda a informação na base de dados teve que ser acrescida do mesmo campo. Para que o valor do utilizador possa ser lido, uma vez mais foi necessário retomar a sessão actual com a função *session_start()* e acrescentar na *query* sql com os campos a introduzir na tabela o valor contido na variável de sessão "logado".

```
session_start();

(...)

mysql_query("INSERT INTO ‘basedados’.’upload’ (‘nome’, ‘uploader’, ‘coment’, ‘data’) VALUES (‘". $ficheiro["name"] ."’, ‘". $_SESSION["logado"] ."’, ‘". $comentario ."’, ‘". $data ."’)");
```

Depois de introduzido o nome do utilizador que efectuou o upload na base de dados, falta dar feedback do mesmo quando se listam os ficheiros no servidor. Para esse efeito foi novamente editada a página "listar.php". A estrutura da tabela gerada teve que suportar mais uma célula para o nome do uploader e foi nesta página que se diferençaram os privilégios dos diferentes utilizadores. Convencionou-se que o utilizador "empresa" funcionaria como um administrador, pleno de privilégios, enquanto que o utilizador "cliente" teria os mesmos privilégios excepto o de apagar ficheiros remotamente. Desta forma todos os utilizadores podem fazer uploads, listar e aceder aos ficheiros, mas apenas utilizadores autenticados como "empresa" podem retirar ficheiros do servidor. Esta distinção foi posta em pratica na página de listagem onde, após criado o cabeçalho da tabela, é efectuado um teste pelo tipo de utilizador autenticado. Se o utilizador for identificado como "cliente" a célula onde inicialmente aparecia a opção para apagar o ficheiro não aparece visível, enquanto que se o utilizador for identificado como "empresa" a estrutura se mantém inalterada face ao protótipo anterior.

```
echo "<div align=right><br> <a href='main.php'> voltar </a></div>";
echo "<table border=10 width=100%>";

if ($_SESSION[‘logado’] == "empresa") {
  echo "<tr> <td align=’center’> <b> nome </b> </td> <td align=’center’> <b>
apagar </b> </td> <td align=’center’> <b> uploader </b> </td> <td
align=’center’> <b> comentario ao upload </b> </td> <td align=’center’> <b>
upload feito em </b> </td> </tr>";
(...)
} else {
  echo "<tr> <td align=’center’> <b> nome </b> </td> <td align=’center’> <b>
uploader </b> </td> <td align=’center’> <b> comentario ao upload </b> </td>
<td align=’center’> <b> upload feito em </b> </td> </tr>";
(...)
}
```

Os utilizadores têm ainda que ter disponível a opção de terminar a sessão em segurança e, para esse efeito, foi inserido ainda na página principal (“main.php”) um link simples para a página “logout.php”, que foi desenvolvida de seguida. Já na página de logout, depois de novamente retomada a sessão com a função *session_start()*, foi usada uma estrutura de decisão simples para verificar se a variável “logado” tem algum valor com a função php *isset()* e, em caso afirmativo, passa o valor para nulo (NULL), com a função *unset()* e o utilizador é de novo enviado para a página de login.

```
session_start();

if (isset($_SESSION[‘logado’])) {
  unset($_SESSION[‘logado’]);
}
header(‘location: login.php’);
```

O outro requisito funcional definido centrava-se na restrição no upload de determinados tipos de ficheiros para o servidor, assim como ficheiros com tamanhos considerados demasiado pesados. O controlo de tamanho dos ficheiros ainda foi desenvolvido, mas foi acabou por ser abandonado no protótipo final, uma vez que depois de alguns testes considerou-se que esta opção poderia limitar negativamente o uso da plataforma na partilha de, por exemplo, vídeos em *raw* ou documentos Photoshop mais pesados. Também a opção de limitar alguns tipos de ficheiros não foi devidamente pensada e acabou por ser abandonada em conjunto com a anterior. No entanto, para limitar o upload dos ficheiros, tanto em tamanho como por tipo, foi usada uma estrutura de decisão simples que testava o ficheiro proveniente do formulário, ainda antes de o enviar para o servidor. O código que se segue é um exemplo da lógica usada, antes de

ser abandonada, neste caso especifico a limitar ficheiros do tipo zip ou ficheiros maiores que, aproximadamente 2Mb (~1,91Mb), uma vez que os valores são em bytes.

```
if       ((!$_FILES["file"]["type"]       ==       'application/zip')       ||
($_FILES["file"]["size"] < 2000000)) {
```

Falta ainda referir que, à semelhança do que se passava com o protótipo da *newsletter*, em todas as páginas de processamento foi inserido um *header* que, passado um determinado intervalo de tempo, reencaminha o utilizador automaticamente para a página onde se encontrava anteriormente, tipicamente a página "main.php".

```
header("refresh:5;url=main.php");
```

d. Validação

Como já foi referido anteriormente, a validação dos protótipos desenvolvidos esteve a cargo, inicialmente, apenas dos orientadores e gestores de projectos; já só numa fase final foram entregues para a realização de testes em *focus groups*. A validação inicial de cada protótipo decorreu individualmente, tendo sido realizada após o fase de prototipagem dos mesmos. Para esta primeira etapa, as funcionalidades dos protótipos foram cruzadas com os requisitos funcionais de modo a testar a validade de cada um deles individualmente. Apenas se o resultado final coincidisse com os resultados esperados o protótipo era considerado válido e o investigador poderia prosseguir no estudo.

Uma vez que não se registaram imprevistos nem dificuldades de maior no decorrer deste estudo, no dia marcado para a entrega dos protótipos estes cumpriam sempre com o que tinha sido acordado anteriormente, excepção feita apenas no caso do segundo protótipo, cujo requisito de limitação dos uploads foi abandonado justificadamente. Terminado o ciclo de desenvolvimento de ambos os protótipos, estes foram colocados num servidor público online e toda a informação relativa ao acesso a ambos os serviços foi compilada e disponibilizada à equipa de programadores residente da Dreamlab para efectuarem uma nova fase de testes. A equipa de testes compreendeu entre 6 a 8 pessoas, número suficiente de elementos para a realização de este tipo de testes uma vez que, segundo Nielsen (2000), os primeiros 5 participantes conseguem detectar 80% dos problemas de usabilidade dos produtos multimédia. Foi solicitado à equipa de testes que procurassem por eventuais erros, falhas ou fragilidades, tendo em

conta os requisitos funcionais definidos anteriormente. Foi ainda conferida aos elementos com compuseram a equipa de testes a liberdade de deixarem opiniões ou sugestões na secção de comentários da plataforma de upload de ficheiros ou então pelo envio de e-mails. Os testes decorreram ao longo de uma semana, período após o qual foram comunicados os resultados ao investigador.

e. Análise crítica dos protótipos desenvolvidos

Dentro da temática desta investigação, os módulos foram escolhidos para desenvolvimento tendo em conta não só as funcionalidades acordadas mas também por permitirem o acréscimo de outras novas posteriormente. Esta característica confere assim alguma plasticidade aos módulos, podendo ser usados em diferentes contextos e não se limitando apenas ao estado actual. Uma das grandes vantagens inerentes a uma aplicação deste género passa por actualizações futuras, ou seja, após construída uma base funcional, em qualquer altura poderão ser acrescentadas funcionalidades que na altura se considerem relevantes, de modo a melhorar o serviço prestado pelo módulo, a qualquer dada altura. Assim, o nível de desenvolvimento de ambos consistiu apenas na criação de uma base que garantisse o funcionamento a um nível pouco profundo, com requisitos funcionais relativamente simplistas.

Como já foi referido anteriormente, os requisitos funcionais acordados em conjunto entre o investigador e o gestor de projectos da empresa foram cumpridos satisfatoriamente, no entanto, também por ser tratarem apenas de protótipos estes têm algumas limitações. De um ponto de vista expressamente técnico, o código gerado não está completamente optimizado, havendo repetições que podem ser contornadas recorrendo, por exemplo, à implementação da instrução *include()*. Um caso recorrente nos protótipos desenvolvidos é a tentativa de ligação à base de dados, expressão essa que se repete várias vezes, sempre para o mesmo efeito, quando podia ter sido criada uma página php dedicada com as instruções necessárias para garantir a comunicação com a base de dados e depois incluída sempre que necessário, nas restantes páginas.

Ainda na comunicação com a base de dados sql é de referir outro pormenor que poderia precisar de revisão futura que se prende com a criação da própria base de dados inicial e, consecutivamente, com as respectivas tabela onde vão ser guardados tanto os contactos como a informação relativa ao upload de ficheiros. No caso dos protótipos desenvolvidos, sempre que é necessário dar entrada de dados na tabela dedicada é forçada a criação da estrutura da base de dados de destino completa, em vez de se

realizar uma *query* à base de dados para saber se essa estrutura já existe ou não. Deste modo, há a garantia de que o protótipo só segue em frente depois de devidamente criada a estrutura necessária e o resultado final acaba por ser o mesmo em termos práticos; no entanto, uma verificação inicial antes de forçar a criação da tabela teria sido mais correcta.

Especificamente no caso da *newsletter*, também a funcionalidade de cancelar subscrições não está optimizada, uma vez que foram criadas duas páginas para o efeito que diferem apenas numa instrução: o método de entrada do contacto. As páginas criadas servem propósitos diferentes, sendo que uma é para apagar contactos através de uma eventual página de gestão dos contactos e a outra é acedida através de um link que é gerado em cada email enviado.

Também na funcionalidade de cancelar subscrições, quando o email a apagar é inserido manualmente, não é efectuado nenhum teste quanto à sua validade como acontece, por exemplo, ao dar entrada de um email novo na base de dados. Uma vez mais, o resultado final é indistinguível, tendo em conta que os contactos só são retirados da base de dados se for encontrado um exactamente igual ao introduzido.

Ainda relativamente à opção de cancelar contactos, não existe um pedido de confirmação, o que significa que se for usada através do link presente nos emails enviados é possível apagar definitivamente um contacto apenas com um clique, intencional ou por engano.

Outro teste que se pode incluir posteriormente é na inserção de um novo contacto, já que não é verificada a existência prévia do contacto na base de dados, o que pode culminar em registos duplicados. Havendo registos duplicados, o que vai acontecer é que se for enviada a *newsletter*, esse contacto vai receber o email em duplicado, ou então, ao tentar apagar este contacto da base de dados, são apagadas todas as entradas, repetidas ou não.

Finalmente, relativamente aos anexos, o protótipo apenas permite a inclusão de ficheiros no formato JPG, e apenas de um anexo de cada vez. Esta decisão foi consciente, uma vez que é preciso indicar no *header* dos emails em formato MIME qual o tipo de ficheiro que vai ser enviado e, para fins de protótipo, um tipo de ficheiro era suficiente para demonstrar a ferramenta a funcionar. Uma implementação futura necessita que seja efectuado um teste ao ficheiro introduzido no formulário o altere o *header* da mensagem dinamicamente, de modo a permitir a inclusão de qualquer anexo, ou até mesmo mais que um anexo em simultâneo. Pode ainda ser acrescentada a opção

de incluir texto no email final, algo que ficou de fora do protótipo uma vez que a sua introdução era imediata e não trazia nada de novo a este estudo.

Relativamente à plataforma de partilha de ficheiros, além das questões de optimização de código referidas anteriormente e que abrangem os dois protótipos, nesta fase é identificável apenas uma limitação especifica. Durante o processo de upload de um ficheiro para o servidor remoto, não há qualquer verificação da existência da pasta de destino necessária, o que obriga a uma preparação prévia do servidor anteriormente à utilização do módulo. Este passo só tem que ser realizado uma vez; no entanto, o próprio módulo deveria prever esse tipo de situação e, caso a pasta “files” não exista no servidor, criá-la antes de proceder ao upload dos ficheiros. No ponto em que está o protótipo, se a pasta não existir no servidor, a operação de upload é aparentemente completada com sucesso e toda a informação é introduzida na base de dados correctamente, no entanto o ficheiro não é guardado remotamente.

Além desta limitação, é ainda de referir, novamente, que foi abandonado um dos requisitos funcionais na fase de desenvolvimento do protótipo, definido anteriormente. Como já foi explicado neste documento, a decisão foi tomada após reflexão ponderada, de modo a não limitar o uso do protótipo, pelos menos não nesta fase. Se, no entanto, numa futura implementação deste módulo for considerado que um controlo semelhante ao previsto inicialmente é benéfico ao uso da plataforma, a solução a implementar também já foi exposta neste documento e é de rápida introdução.

Outro pormenor que deve ser mencionado, prende-se com o acesso às páginas individualmente: para aceder às páginas “main.php”, “listar.php” e “upload.php” é necessária a autenticação do utilizador, estado este que fica guardado numa variável de sessão que vai acompanhado o desenrolar da plataforma, caso contrário remete o utilizador automaticamente para a página de login. No caso das restantes páginas, como não necessitam de autenticação no protótipo desenvolvido, podem ser acedidas directamente; no entanto, como falta a informação que normalmente seria passada de passos anteriores, estas não processam nada e passados 5 segundos reencaminham o utilizador para a página principal. De futuro podem ainda ser acrescentadas a este modulo funcionalidades tais como a opção de listar os ficheiros por uploader, ou ainda organizar os ficheiros por ordem alfabética, ascendente ou descendente. Tendo a base do módulo já sido desenvolvida com sucesso, estas funcionalidades ou outras podem ser facilmente acrescentadas, conforme a utilização futura assim o considere necessário ou vantajoso no uso da ferramenta.

Quando comparados, a clara discrepância de limitações entre ambos os protótipos é também um reflexo do tempo dispendido para cada um deles, assim como da importância relativa da plataforma de partilha face à *newsletter*. É ainda de referir que, apesar das limitações actuais, os protótipos encontram-se em funcionamento e asseguram os serviços mínimos propostos sem falhas e em qualquer momento podem ser implementados ou acrescidos de novas funcionalidades.

f. Proposta de modelo de representação algorítmica

Os protótipos realizados baseiam-se na comunicação entre diversas páginas dedicadas e uma base de dados comum a ambos, mas com tabelas diferentes criada especificamente para cada um dos casos. Ainda relativamente à base de dados, o utilizador tem que ter privilégios de escrita, para o correcto funcionamento dos módulos em ambos os casos. Se, em qualquer dos módulos, a ligação à base de dados falhar, não há como avançar além das páginas de entrada.

No caso do protótipo da *newsletter*, este consistiu numa página inicial em formato html, onde eram disponibilizados todos os formulários e links necessários para aceder a todas as funcionalidades requeridas e, ainda, cinco páginas php para cada uma das funções individualmente. Em todos os casos, exceptuado a página “cancelar2.php”, a comunicação entre páginas é feita através do método *http post*, de modo a que não sejam visíveis dados na barra de endereços. Estas 5 páginas em php comunicam com a base de dados mysql, mais especificamente com a tabela “contactos”, criada de propósito para o efeito, e realizam diferentes operações. A página “inserir.php” adiciona novos valores aos já existentes na base de dados, especificamente, adiciona o valor introduzido no formulário anterior como um contacto novo. A página “listar.php” faz uma pesquisa e retorna os valores encontrados, que neste caso serão todos os contactos existentes base de dados nesse momento. A página “cancelar1.php” é acedida através de um formulário presente na página inicial onde, depois de introduzido um contacto é realizada uma pesquisa à base de dados e são eliminados todos os valores que coincidirem com os introduzidos pelo utilizador. A página “enviar.php” recebe os campos “assunto” e “anexo” provenientes do formulário anterior e efectua uma pesquisa à base de dados, da qual são retirados todos os contactos existentes. É depois processado um email individual para cada contacto existente na base de dados, onde é introduzido o assunto e anexado o ficheiro anteriormente escolhido e depôs enviado ainda com um link para a página

“cancelar2.php”, que realiza uma operação em tudo igual à anterior página de cancelamento de subscrição. A única diferença é no método de entrada dos dados, uma vez que nesta página vai-se buscar a informação directamente ao link, pelo método *get*.

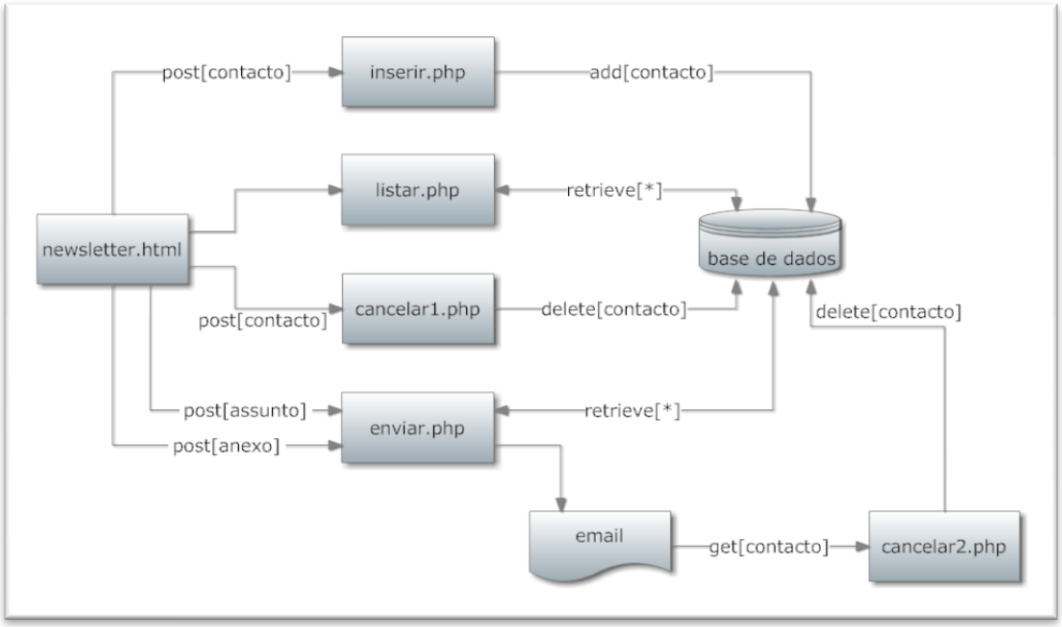

Ilustração 4 - Representação visual do protótipo de *newsletter*

Já o protótipo da plataforma de partilha de ficheiros é constituído integralmente por páginas desenvolvidas em php, seis mais especificamente. A página principal, “main.php” só está disponível após realizada a autentificação do utilizador, que fica guardada numa variável de sessão que, além desta página, é também necessária nas páginas upload e listar. Tal como acontecia anteriormente, o método de envio de informação entre páginas usado predominantemente é o *post*, para que esta não seja visível através dos links. Na página “main.php”, após preenchido o formulário para upload de um ficheiro novo, a informação é processada na página “upload.php” que adiciona na tabela “upload” o nome do ficheiro e o comentário associado, provenientes do formulário, a data actual retirada dinamicamente do relógio interno do servidor e finalmente a informação presente na variável de sessão relativa ao estado de login, para diferenciação entre os utilizadores. Na página “listar.php”, para que possam ser apresentados todos os ficheiros presentes no servidor remoto, assim como a informação individual de cada um, é realizada uma nova ligação à base de dados seguida de uma *query* que vai retornar de forma ordenada todos os valores presentes na tabela da base de dados mysql. Quando listados todos os ficheiros é criado ainda na tabela de *output* um link para poder apagar

cada um individualmente, que envia o nome do ficheiro como parâmetro. Ao aceder a este link, a informação é processada na página “apagar.php” que vai buscar o nome do ficheiro ao link com o método *get* e realiza uma nova *query* à base de dados e apaga o respectivo ficheiro. A opção de efectuar um logoff só está disponível na página principal, onde é também possível consultar qual das sessões está iniciada e, se escolhida, o processamento é efectuado na pagina “logout.php” que verifica a validade da variável de sessão anteriormente criada e caso esta se verifique, limpa-a e reencaminha o utilizador para a página de login.

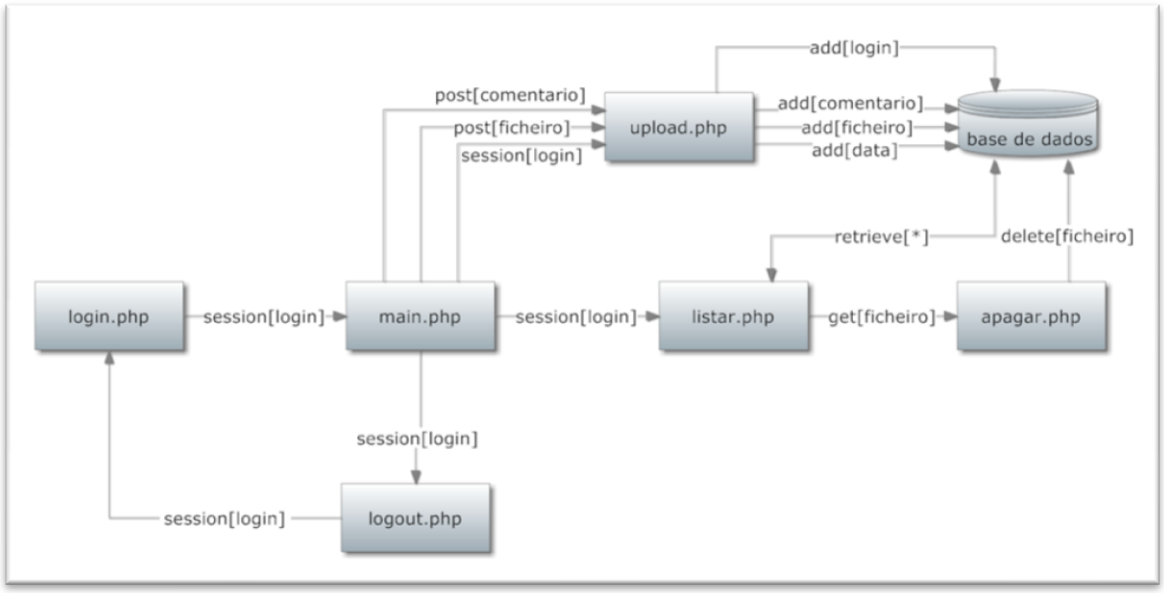

Ilustração 5 - Representação visual do protótipo da plataforma de partilha de ficheiros

IV. Conclusões

1. Limitações do estudo

As limitações destes estudo podem ser divididas em duas categorias distintas, ainda que inter-dependentes: limitações de cariz temporal e de cariz cientifico.

Este estudo foi realizado no âmbito de uma dissertação de mestrado e, por isso mesmo, está limitado aos intervalos de tempo estipulados para o efeito. Uma consequência mais imediata deste factor foi o reduzido numero de protótipos desenvolvidos. Uma vez que o estudo envolvia uma primeira parte mais exploratória, não só das necessidades e práticas da empresa, como da própria tecnologia a usar para o desenvolvimento, terminado o intervalo de tempo dedicado ao estágio empresarial, apenas foi possível o desenvolvimento de dois protótipos. Apesar destes terem sido estudados e pensados de forma a serem o mais úteis possível para a empresa, dado um período de tempo maior poderiam ter sido prototipados outros módulos, que abrangessem outras áreas que não apenas a comunicação entre empresas e clientes.

Os dois protótipos permitem a respostas à questão de investigação, no entanto, quantos mais módulos tivessem sido prototipados, mais valioso poderia ter sido o contributo deste trabalho para a empresa e para a temática em geral. O factor tempo foi também influenciado por claras limitações de cariz cientifico, uma vez que, anteriormente a este estudo o investigador nunca tinha trabalhado com a tecnologia php. Este factor implicou uma aprendizagem à medida que se iam prototipando os módulos, o que constituiu uma clara limitação, mas também uma oportunidade de abordar o problema prático de uma forma completamente nova e com um enfoque apenas no essencial esquecendo o acessório, pelo menos nesta fase.

Outra limitação ocorreu ainda na fase de acompanhamento, anterior à prototipagem dos módulos, onde foi acompanhado o desenvolvimento de diversos projectos a serem realizados nesse período na empresa; no entanto, estes já decorriam antes da chegada do investigador, o que ,aliado ao tempo calendarizado para o efeito, impediu o que fosse observado o ciclo de desenvolvimento completo para, pelo menos, um projecto multimédia. Também na fase de observação poderiam ter sido realizadas mais entrevistas, facto que na altura não se considerou particularmente relevante.

2. *Perspectivas de trabalho futuro*

Tendo este estudo por base uma proposta de optimização das lógicas de trabalho e uma vez que o feedback recebido relativamente aos protótipos desenvolvidos foi positivo, a elaboração de trabalho futuro deverá começar por eliminar algumas das fraquezas apontadas aos protótipos no capítulo de análise critica dos mesmos. Ainda que os módulos estejam a funcionar à data de conclusão deste estudo, os protótipos desenvolvidos devem ser revistos antes de serem aplicados comercialmente por equipas especializadas no sentido de optimizar o seu código fonte e garantir um serviço sem a ocorrência de *bugs*. A implementação de novas funcionalidades sobre os protótipos desenvolvidos é também uma realidade, uma vez que garantindo os serviços mínimos, ambos estão prontos a serem acrescidos de todas as características que valorizem a sua utilização futura.

Devem ainda ser realizadas novas entrevistas de modo a perceber que outros módulos possam servir como uma mais valia para a empresa, para ser posteriormente desenvolvidos. Pode ainda haver novo acompanhamento de projectos e análise dos projectos anteriormente desenvolvidos pela Dreamlab, para realçar quais os serviços mais recorrentes para que possam ser desenvolvidos de forma modular. Uma lista de novos módulos pode ainda surgir de interesses da própria empresa, tal como aconteceu especificamente no caso da plataforma de partilha de ficheiros prototipada.

Finalmente, perspectiva-se que antes de serem usados os protótipos desenvolvidos, estes devem ainda ser alvo de uma fase de testes de usabilidade junto de eventuais clientes ou uma amostra cuidadosamente escolhida, ao contrário dos testes efectuados apenas em *focus groups* no decorrer desta investigação. Uma vez que os protótipos desenvolvidos se focaram nas funcionalidades do ponto de vista técnico, estes não eram de todo representativos da apresentação e grafismo finais dos módulos devidamente implementados. Após devidamente desenvolvido o grafismo dos módulos, estes devem ser sujeitos a testes que garantam maximizar a sua funcionalidade e não apenas a garantia de operabilidade do ponto de vista técnico.

3. Conclusões

O estudo decorreu no âmbito do desenvolvimento de serviços especificamente para a empresa Dreamlab, sediada em Aveiro. No entanto, o resultado final foi a prototipagem de dois serviços típicos que podem ser implementados numa multiplicidade de situações, desde sites pessoais ou fins comerciais. O estudo realizado insere-se assim dentro da temática da agilidade de desenvolvimento de conteúdos Web, um tema que apesar de não ser novo, mantém-se actual nos dias de hoje (Cockburn, 2006).

A questão de partida desta investigação perguntava “como optimizar o processo de desenvolvido de sites Web?” e a resposta surge através da implementação de serviços, de um modo que possam ser aplicado repetidas vezes, para diferentes pedidos por parte dos clientes da empresa.

O estudo realizado visava a conceptualização, desenvolvimento e teste de módulos capazes de vir a ser úteis no processo de desenvolvimento de sites Web, na Dreamlab. Para que fosse possível dar resposta aos objectivos do estudo, foi inicialmente realizado uma análise sobre que serviços seriam do interesse da empresa ser prototipados. A partir desta análise foram criados objectivos específicos com resultados mensuráveis, que foram atingidos com sucesso, não só dentro do contexto do desenvolvimento de sites Web na Dreamlab, mas também alargáveis à temática geral do desenvolvimento ágil e modularidade.

Apesar da especificidade operacional deste estudo, entendemos que, de forma consistente com os princípios do desenvolvimento ágil, é possível reutilizar os resultados obtidos para construir soluções potencialmente aplicadas a outros contextos.

Referências Bibliográficas

Abrahamsson, Pekka et al (2002) *Agile software development methods: review and analysis*, VTT

Arsanjani, Ali (2004) *Service-oriented modeling and architecture*, http://www.ibm.com/developerworks/webservices/library/ws-soa-design1/ (15-06-2010)

Arsanjani, Ali et al (2007) *Design an SOA solution using a reference architecture*, http://www.ibm.com/developerworks/library/ar-archtemp/ (15-06-2010)

Baldwin, Carliss Y e Clark, Kim B. (1999) *Design rules: the power of modularity*, MIT Press

Baldwin, Carliss Y. (2007) *Frameworks for thinking about modularity, industry architecture and evolution*

Baldwin, Carliss Y. e Clark, Kim B. (1997) *Managing in the modular age*, Blackwell Publishing

Bass, Len et al (2003) *Software architecture in practice*, Addison Wesley

Batory, Don e O'Malley Sean (1992) *The design and implementation of hierarchical software systems with reusable components*

Beck, Kent (1999) *Extreme programming explained*, Addison Wesley

Cai, Yuanfang e Sullivan, Kevin J. (s.d.) *A value-oriented theory of modularity in design*

Clifton, Marc (2003) *What is a framework?*, http://www.codeproject.com/KB/architecture/WhatIsAFramework.aspx (15-06-2010)

Cockburn, Alistair (2006) *Agile software development: the cooperative game, second edition*, Addison Wesley

Cottenier, Thomas e Elrad, Tzilla (s.d.) *Validation of context-dependet aspect-oriented adaptations to components*

Curtis, Nathan (2009) *Modular web design*, Peachpit Press

Doddavula, Shyam Kumar e Karamongikar, Sandeep (2005) *Designing and enterprise application framework for service oriented architecture*, InfoSys

Erl, Thomas (2005) *Service-oriented architecture: concepts, technology and design*, Prentice Hall PTR

Fernandes, João Miguel e Machado, Ricardo Jorge (s.d.) *Projecto de hardware digital orientado por objectos*

Fowler, Martin (2005) *ServiceOrientedAmbiguity*, http://www.martinfowler.com/bliki/ServiceOrientedAmbiguity.html (15-06-2009)

Fowler, Martin (2005) *The new methodology*, http://www.martinfowler.com/articles/newMethodology.html (15-06-2009)

Gaedke, Martin e Gräf, Guntram (2000) *Development and evolution of web-applications using the webcomposition process model*

Glissom, William Bradly et al (s.d.) *Web development evolution: the business perspective on security*

Grosskurth, Alan e Godfrey, Michael W. (2006) *Architecture and evolution of the modern web browser*

Highsmith, Jim (2004) *Agile project management: creating innovative products*, Addison Wesley

Hill, Janelle B. et al (2007) *Magic quadrant for business process management suites 2007*, Gartner Inc

Hsuan, Juliana (1998) *Modularization in black-box design: implications for supplier-buyer partnerships*

Huvar, Martin et al (2008) *Developing applications with enterprise SOA*, Galileu Press

Jacobson, Ivar e Ng, Pan-Wein (2004) *Aspect-oriented software development with use cases*, Addison Wesley

McClamrock, Ron (s.d.) *Modularity*

McConnell, Steve (1996) *Rapid Development: taming wild software schedules*, Microsoft Press

Meirelles, Junia Cristina J.P. e Moura, Mônica (2007) *Web 2.0 novos paradigmas projetuais e informacionais*, InfoDesign

Natis, Yefim V. (2003) *Service-oriented Architecture Scenario*, Gartner

O'Leary, Ciarán et al (2006) *Collaborative online development of modular intelligent agents*, Ercim News

Paige, Richard et al (2002) *Combining agile practices with UML and EJB: a case study in agile development*

Quivy, Raymond e Campenhoudt, Luc Van (2008) *Manual de Investigação em Ciências Sociais*, Gradiva

Reitman, Laurel et al (2007*) Service oriented architecture and specialized messaging patterns*, Adobe

Ribeiro, Márcio de Medeiros (2008*) Restructuring test variables in software product lines*, Tese de Mestrado

Rinard, Martin et al(2004) *A classification system and analysis for aspect-oriented programs*

Rotem-Gaz-Ol, Arnon (s.d.) *What is SOA anyway?*

Sant'Anna, Cláudio Nogueira (2008*) On the modularity of aspect-oriented design: a concern-driven measurement approach*, Tese de Doutoramento

Schattkowsky, Tim e Lohmann, Marc (s.d.) *Rapid development of modular dynamic web sites using UML*

Senger, Vinicius (2008) *Modularidade com Java module sustem & OSGi*, Global Code

Silva, Heriberto do Ouro Lopes (2007) *Modularidade no desenvolvimento do produto*

Silva, Luis Moura e Buyya, Rajkumar (s.d.) *Parallel programming models and paradigms*

Soares, Sérgio Castelo Branco (2004) *An aspect-oriented implementation method*, Teste de Doutoramento

Sommerville, Ian (2006) *Software engineering 8th* edition, Wesley Addison

Thomas, Dave e Hansson, David Heinemeter (2006) *Agile web development with Rails*, The pragmatic bookshelf

Watkins, John (2009) *Agile testing: how to succeed in an extreme testing environment*, Cambridge University Press

Wampler, Dean (s.d.) *Aspect-oriented design principles: lessons from object-oriented design*

Wampler, Dean (2007) *Aspect oriented programming in academia and industry*

Wampler, Dean (2007) *Aspect oriented programming and design for Java and AspectJ*

Wampler, Dean (2003) *The future of aspect oriented programming*

Whitworth, Elizabeth (2006) *Agile experience: communication and collaboration in agile software development teams,* Tese de Mestrado

Windley, Phillip J. (2006) *SOA Governance: rules of the game*, Inforword.com

Anexos

1. *Código fonte*

a. Protótipo de *newsletter*

i. Newsletter.html

```
<html>
<!--
pagina meramente exemplificativa
so interessam os formulários
-->

<head> <title> newsletter </title> </head>
<body>
  <table border=0 width='500'>
    <tr> <td colspan=2>   </td></tr>
    <tr>
      <form method=post action='inserir.php'>
        <td>
          adicionar subscricao nova
        </td>
        <td colspan=2>
          <input type=text name='contacto'>
          <input type=submit value='ok'>
        </td>
      </form>
    </tr>
    <tr> <td colspan=2>   </td></tr>
    <tr> <td colspan=2>   </td></tr>
    <tr>
      <form method=post action='cancelar1.php'>
        <td> cancelar uma subscricao existente </td>
        <td>
          <input type=text name='contacto'>
          <input type=submit value='ok'>
        </td>
      </form>
    </tr>
    <tr> <td colspan=2>   </td></tr>
    <tr> <td colspan=2>   </td></tr>
    <tr>
      <td colspan=2> enviar newsletter </td>
    </tr>
    <tr>
      <form method='post'action=enviar.php enctype='multipart/form-data'>
        <td> inserir assunto </td>
        <td>
          <input type=text name='assunto'>
        </td>
      <tr>
        <td> escolher anexo (!!JPG!!) </td>
        <td>
```

```
          <input type=file name='ficheiro'>
        </td>
      </tr>
      <tr>
        <td colspan=2>
          <input type=submit value='enviar'>
          <input type=reset value='cancelar'>
        </td>
      </form>
    </tr>
    <tr> <td colspan=2>   </td></tr>
    <tr> <td colspan=2>   </td></tr>
    <tr>
      <td colspan=2>
        <a href="listar.php">listar contactos existentes</a>
      </td>
    </tr>
    <tr> <td colspan=2>   </td></tr>
  </table>
</body>
</html>
```

ii. Listar.php

```
<?php
//lista todos os contactos existentes na base de dados

  header("refresh:5;url=newsletter.html");
  $contactos = 0;
  $ligacao  =  mysql_connect("localhost","root");  //dados  da  ligacao  podem
precisar de alteracao
  if (!$ligacao)
  {
    die ("nao foi possivel efectuar a ligacao <br>" . mysql_error() . "<br>");
  }
  mysql_select_db(basedados, $ligacao);
  $query = mysql_query("SELECT * FROM `listagem`");
  //percorre todos os registos na tabela e mostra contacto
  while ($row = mysql_fetch_assoc($query))
  {
    echo $row['contacto'] . "<br>";
    $contactos++;
  }
  echo "<br> de momento existem " . $contactos . " contactos validos";
?>
```

iii. Inserir.php

```
<?php
//inscricao dos mails na base de dados
//entrada dos dados vindos do formulario de origem com o nome "contacto"
```

```
  header("refresh:5;url=newsletter.html");
  $contacto = $_POST["contacto"];
  $valido = 1;
  //se o contacto for invalido, mostra aviso e nao envia para a base de dados
  if(!stristr($contacto,"@") OR !stristr($contacto,"."))
  {
    $valido = 0;
    echo "e-mail invalido <br>";
  }
  //se o contacto for valido, guarda na base de dados
  if ($valido == 1)
  {
    //testa ligagacao a base de dados e cria base de dados nova
    $ligacao = mysql_connect("localhost","root"); //dados da licacao podem
precisar de alteracao
    if (!$ligacao)
    {
      die ("nao foi possivel efectuar a ligacao <br>" . mysql_error() .
"<br>");
    }
    if (mysql_query("CREATE DATABASE basedados",$ligacao))
    {
      //cria tabela nova
      $tabela = "CREATE TABLE `basedados`.`listagem` (`contacto` VARCHAR(25)
NOT NULL) ENGINE = MyISAM";
      mysql_query($tabela);
      echo "base de dados criada com sucesso <br>";
    }
    else
    {
      //echo "nao foi possivel criar a base de dados <br>" . mysql_error() .
"<br>";
    }
    //guarda contacto na tabela
    $inserir = "INSERT INTO `basedados`.`listagem` (`contacto`) VALUES ('".
$contacto ."')";
    mysql_query($inserir);
    echo "obrigado pela subscricao <br>";
    mysql_close($ligacao);
  }
  //echo $inserir;
?>
```

iv. Enviar.php

```
<?php
//envio do mail para todos os contactos na base dados
//entrada dos dados vindos do formulario de origem com o nome "assunto" e
"mensagem"
//campos "assunto" e "anexo" obrigatorios

  header("refresh:5;url=newsletter.html");
  $assunto = $_POST["assunto"];
  $ficheiro = $_FILES["ficheiro"]["name"];
```

```
  $anexo = $_FILES["ficheiro"]["tmp_name"];
  $contactos = 0;
  $bound_text = "boundry generico"; //para alterar
  $bound = "--". $bound_text. "\r\n";
  $bound_last = "--". $bound_text. "--\r\n";
  //testa o campo assunto
  if ($assunto=="")
  {
    echo "assunto em branco";
  }
  else
  {
    //testa o campo anexo
    if ($ficheiro=="")
    {
      echo "nao foi escolhido um anexo";
    }
    else
    {
      //testa ligaçao a base de dados
      $ligacao = mysql_connect("localhost","root"); //dados da ligacao podem
precisar de alteracao
      if (!$ligacao)
      {
        die ("nao foi possivel efectuar a ligacao <br>" . mysql_error() .
"<br>");
      }
      mysql_select_db(basedados, $ligacao);
      $query = mysql_query("SELECT * FROM `listagem`");
      //abre o anexo para leitura
      //le e codifica para envio
      $abre = fopen($anexo, "rb");
      $anexo = fread($abre, filesize($anexo));
      $anexo = chunk_split(base64_encode($anexo));
      fclose($abre);
      //definicao dos headers da mensagem
      $headers = "From: <teste>\n"; //para alterar
      $headers .= "MIME-Version: 1.0\n";
      $headers .= "Content-Type: multipart/mixed; boundary=\"$bound_text\"";
      //percorre todos os registos na tabela e envia mail
      while ($row = mysql_fetch_assoc($query))
      {
        //definicao do corpo da mensagem
        $mensagem .= "Content-Type: text/html; charset=\"iso-8859-1\"\r\n";
        $mensagem .= "Content-Transfer-Encoding: 7bit\r\n\r\n";
        $mensagem     .=     "\r\n\r\n     <a     href=cancelar2.php?contacto=".
$row['contacto'] ."> clique aqui para cancelar a subscricao </a>"; //link pode
precisar de alteracao
        $mensagem .= $bound;
        $mensagem .= "Content-Type: image/jpg; name=" . $ficheiro ." \r\n";
        $mensagem .= "Content-Transfer-Encoding: base64 \r\n";
        $mensagem  .= "Content-disposition: attachment; file=" . $ficheiro ."
\r\n";
        $mensagem .= $anexo;
        $mensagem .= $bound_last;
        //envia mail para todos os contactos da lista
        if (mail($row['contacto'], $assunto, $mensagem, $headers))
```

```
        {
          echo "newsletter enviada com sucesso para " . $row['contacto'] .
"<br>";
          $contactos++;
        }
        else
        {
          echo "falha no envio para " . $row['contacto'] . "<br>";
        }
      }
      echo "<br> newsletter enviada com sucesso para " . $contactos . "
contactos";
      mysql_close($ligacao);
    }
  }
?>
```

v. Cancelar1.php

```
<?php
//cancelar uma subscricao existente
//entrada dos dados vindos do formulario de origem com o nome "cancelar"
//contacto de mail tem q vir completo

  header("refresh:5;url=newsletter.html");
  $existe = 0;
  $cancelar = $_POST["contacto"];
  $ligacao = mysql_connect("localhost","root"); //dados da licacao podem
precisar de alteracao
  if (!$ligacao)
  {
    die ("nao foi possivel efectuar a ligacao <br>" . mysql_error() . "<br>");
  }
  //procura contacto na base de dados
  mysql_select_db(basedados, $ligacao);
  $query = mysql_query("SELECT * FROM `listagem` WHERE `contacto` LIKE '" .
$cancelar . "'");
  while ($row = mysql_fetch_assoc($query))
  {
    $existe = 1;
  }
  if ($existe==1)
  {
    mysql_query("DELETE FROM `basedados`.`listagem` WHERE `listagem`.`contacto`
= '" . $cancelar . "'");
    echo "o contacto " . $cancelar . " foi retirado da base de dados com
sucesso <br>";
  }
  else
  {
    echo "contacto " . $cancelar . " nao encontrado" . "<br>";
  }
  mysql_close($ligacao);
?>
```

vi. Cancelar2.php

```
<?php
//cancelar uma subscricao existente
//entrada dos dados vindos do link nas newsletters de origem com o nome
"cancelar"
//contacto de mail tem q vir completo

  header("refresh:5;url=newsletter.html");
  $existe = 0;
  $cancelar = $_GET["contacto"];
  $ligacao = mysql_connect("localhost","root"); //dados da ligacao podem
precisar de alteracao
  if (!$ligacao)
  {
    die ("nao foi possivel efectuar a ligacao <br>" . mysql_error() . "<br>");
  }
  //procura contacto na base de dados
  mysql_select_db(basedados, $ligacao);
  $query = mysql_query("SELECT *  FROM `listagem` WHERE `contacto` LIKE '" .
$cancelar . "'");
  while ($row = mysql_fetch_assoc($query))
  {
    $existe = 1;
  }
  if ($existe==1)
  {
    mysql_query("DELETE FROM `basedados`.`listagem` WHERE `listagem`.`contacto`
= '" . $cancelar . "'");
    echo "o contacto " . $cancelar . " foi retirado da base de dados com
sucesso <br>";
  }
  else
  {
    echo "contacto " . $cancelar . " nao encontrado" . "<br>";
  }
  mysql_close($ligacao);
?>
```

b. Protótipo de plataforma de partilha de ficheiros

i. Main.php

```
<?php
  session_start();
  if (!isset($_SESSION['logado']) || $_SESSION['logado'] == "") {
    //verifica estado do login
    header('location: login.php');
    exit;
```

```
  }
?>

<html>
<head> <title> plataforma de partilha de ficheiros </title> </head>
<body>
  sessao iniciada como
  <?php
    if ($_SESSION['logado'] == "empresa") {
      echo  "empresa";
    } elseif ($_SESSION['logado'] == "cliente") {
      echo  "cliente";
    }
  ?> <br>
  <a href="logout.php">terminar sessao</a>
  <br><br><br>
  <form action="upload.php" enctype="multipart/form-data" method="post">
    <br> escolher ficheiro: <br>
    <input type="file" name="ficheiro" size="40">
    <br> adicionar comentario: <br>
    <textarea name='comentario' rows='15' cols='40'></textarea>
    <br>
    <div> <input type="submit" value="enviar"></div>
  </form>
  <a href="listar.php"> listar todos os ficheiros existentes </a>
</body>
</html>
```

ii. Upload.php

```
<?php
//upload de ficheiros para um servidor remoto
//destino na pasta /files

  session_start();
  header("refresh:5;url=main.php");
  $ficheiro = $_FILES["ficheiro"];
  $comentario = $_POST["comentario"];
  $data = date("F j, Y, G:i");
  if ($ficheiro["error"] > 0)
  {
    echo "nenhum ficheiro seleccionado";
  }
  else
  {
    //testa ligagacao a base de dados e cria base de dados nova
    $ligacao = mysql_connect("localhost","root"); //dados da licacao podem
precisar de alteracao
    if (!$ligacao)
    {
      die ("nao foi possivel efectuar a ligacao <br>" . mysql_error() .
"<br>");
    }
    else
```

```
    {
      if (mysql_query("CREATE DATABASE basedados",$ligacao))
      {
        //cria tabela nova
        $tabela = "CREATE TABLE `basedados`.`upload` (`nome` VARCHAR(25) NOT
NULL, `uploader` VARCHAR(25) NOT NULL,`coment` TEXT NOT NULL, `data` TEXT NOT
NULL) ENGINE = MyISAM;";
        mysql_query($tabela);
        echo "base de dados criada com sucesso <br>";
      }
      else
      {
        //echo "nao foi possivel criar a base de dados <br>" . mysql_error() .
"<br>";
      }
      if (file_exists("files/" . $ficheiro["name"]))
      {
        echo "o ficheiro ja existe";
      }
      else
      {
        //guarda o ficheiro na pasta /files
        move_uploaded_file($ficheiro["tmp_name"], "files/". $ficheiro["name"]);
        //guarda nome e comentarios na tabela
        mysql_query("INSERT INTO `basedados`.`upload` (`nome`, `uploader`,
`coment`, `data`) VALUES ('". $ficheiro["name"] ."', '". $_SESSION["logado"]
."', '". $comentario ."', '". $data ."')");
        //feedback para o utilizador
        echo "upload efectuado com sucesso <br>";
        echo "nome do ficheiro: " . $ficheiro["name"] . "<br>";
        echo "comentario: " . $comentario . "<br>";
        echo $data . "<br>";
      }
    }
  }
?>
```

iii. Listar.php

```
<?php
//lista todos os ficheiros e comentarios enviados para o servidor

  $nficheiros = 0;
  session_start();
  if (!isset($_SESSION['logado']) || $_SESSION['logado'] == "")
  {
    //verifica estado do login
    header('location: login.php');
    exit;
  }
  $ligacao = mysql_connect("localhost","root"); //dados da ligacao podem
precisar de alteracao
  if (!$ligacao)
  {
```

```
    die ("nao foi possivel efectuar a ligacao <br>" . mysql_error() . "<br>");
  }
  mysql_select_db(basedados, $ligacao);
  $query = mysql_query("SELECT * FROM `upload`");
  echo "<div align=right><br> <a href='main.php'> voltar </a></div>";
  echo "<table border=10 width=100%>";
  if ($_SESSION['logado'] == "empresa")
  {
    echo "<tr> <td align='center'> <b> nome </b> </td> <td align='center'> <b>
apagar </b> </td> <td align='center'> <b> uploader </b> </td> <td
align='center'> <b> comentario ao upload </b> </td> <td align='center'> <b>
upload feito em </b> </td> </tr>";
    //percorre todos os registos na tabela e mostra resultados
    while ($row = mysql_fetch_assoc($query))
    {
      echo "<tr><td> <a href='files/" . $row['nome'] . "' target='_blank'> " .
$row['nome'] . "</a> </td> <td align='center'> <a href='apagar.php?ficheiro=" .
$row['nome'] . "'> <b> apagar <b> </a> </td> <td> " . $row['uploader'] . "
</td> <td> " . $row['coment'] . " </td> <td> " . $row['data'] . " </td>";
      $nficheiros++;
    }
  }
  else
  {
    echo "<tr> <td align='center'> <b> nome </b> </td> <td align='center'> <b>
uploader </b> </td> <td align='center'> <b> comentario ao upload </b> </td> <td
align='center'> <b> upload feito em </b> </td> </tr>";
    //percorre todos os registos na tabela e mostra resultados
    while ($row = mysql_fetch_assoc($query))
    {
      echo "<tr><td> <a href='files/" . $row['nome'] . "' target='_blank'> " .
$row['nome'] . "</a> </td> <td> " . $row['uploader'] . " </td> <td> " .
$row['coment'] . " </td> <td> " . $row['data'] . " </td>";
      $nficheiros++;
    }
  }
  echo "</table>";
  echo "de momento existem " . $nficheiros . " ficheiros no servidor <br>";
?>
```

iv. Apagar.php

```
<?php
//apaga ficheiros no servidor
//apaga as entradas na base de dados

  header("refresh:5;url=listar.php");
  $ficheiro = $_GET["ficheiro"];
  if (unlink("files/" . $ficheiro))
  {
    $ligacao = mysql_connect("localhost","root"); //dados da ligacao podem
precisar de alteracao
    if (!$ligacao)
    {
```

```
      die ("nao foi possivel efectuar a ligacao <br>" . mysql_error() .
"<br>");
    }
    //procura contacto na base de dados
    mysql_select_db(basedados, $ligacao);
    $query = mysql_query("SELECT *   FROM `upload` WHERE `nome` LIKE '" .
$ficheiro . "'");
    while ($row = mysql_fetch_assoc($query))
    {
      mysql_query("DELETE FROM `basedados`.`upload` WHERE `upload`.`nome` = '"
. $ficheiro. "'");
      echo "o ficheiro " . $ficheiro . " foi apagado com sucesso <br>";
    }
  }
  else
  {
    echo "erro ao apagar o ficheiro";
  }
?>
```

v. Login.php

```
<?php
//form para inicio de sessao

  session_start();
  $erro = '';
  if (isset($_POST['input_login']) && isset($_POST['input_pass'])) {
    //verifica se campos estao preenchidos
    if ($_POST['input_login'] == 'empresa' && $_POST['input_pass'] == 'cenas')
    {
      //inicia sessao
      $_SESSION['logado'] = "empresa";
      header('location: main.php');
      exit;
    } elseif ($_POST['input_login'] == 'cliente' && $_POST['input_pass'] ==
'cenas') {
      //inicia sessao
      $_SESSION['logado'] = "cliente";
      header('location: main.php');
      exit;
    }
    else
    {
      $erro = 'dados errados, tente novamente';
    }
  }
?>

<html>
<head> <title> login </title> </head>
<body>
  <?php
    if ($erro != '') {
```

```
  ?>
  <b><?php echo $erro; ?></b>
  <?php
    }
  ?>
  <form method="post" name="login">
    <table width="250" border="1" cellpadding="2" cellspacing="2">
      <tr>
        <td width="100">login</td>
        <td><input name="input_login" type="text"></td>
      </tr>
      <tr>
        <td width="100">password</td>
        <td><input name="input_pass" type="password"></td>
      </tr>
      <tr>
        <td width="150"> </td>
        <td><input type="submit" name="botao" value="ok"></td>
      </tr>
    </table>
  </form>
</body>
</html>
```

vi. Logout.php

```
<?php
  session_start();
  if (isset($_SESSION['logado'])) {
    //termina a sessao para o administrador
    unset($_SESSION['logado']);
  }
  header('location: login.php');
?>
```

2. Guiões das entrevistas

a. Entrevista elementos da equipa de programação

nome do entrevistado:_____
inserido na dreamlab há:_____

1. **Chegada** de um pedido novo
   a. Quem recebe? _____
   b. Como são normalmente contactados pelos clientes?
      pessoalmente ☐ mail ☐ outro?_____
   c. Quando são efectuados, os pedidos são feitos
      de forma detalhada ☐
      de forma vaga ☐
      acompanhados de rascunhos/storyboards/outros: _____ ☐
   d. Comentários:
      _____
      _____
      _____
      _____

2. **Contacto directo** da equipa de desenvolvimento com o cliente final
   a. Em condições normais? sim ☐ não ☐
   b. Em caso de mudança nas especificações do produto? sim ☐ não ☐
   c. Os envolvidos vão recebendo feedback directo do cliente?
      durante o desenvolvimento ☐
      depois de terminado o produto ☐
      não ☐
   d. Comentários:
      _____
      _____
      _____
      _____

3. **Processamento** de um novo pedido
   a. Como dividem as tarefas? _____
      _____
   b. Pessoas envolvidas? número máximo: _____
      número mínimo: _____
      ordenado médio: _____
   c. O pessoal envolvido é escolhido por que factores?
      estão livres no momento ☐
      conhecimentos específicos ☐
      outros: _____ ☐
   d. A equipa mantêm-se vinculada ao projecto até à sua conclusão?
      sim ☐ não (rotatividade de elementos) ☐

e. Durante o desenvolvimento do projecto ocorrem reuniões conjuntas com todos os envolvidos?
sim, regulares ☐ intervalo médio: _____
sim, ocasionalmente ☐
não ☐

f. Face a um projecto novo, optam por um paradigma de desenvolvimento linear ou definem prioridades?
linear (ordem fixa) ☐
prioridades face aos conteúdos ☐ com enfoque em: _____
_____
_____

g. Na fase inicial do desenvolvimento definem um tempo esperado para conclusão do produto?
sim ☐ com base em: _____
não ☐

h. Qual o tempo médio que demora cada projecto? _____

i. Comentários:
_____
_____
_____
_____

4. Pedidos **tecnologicamente mais desafiantes**

a. Quando assim é necessário, é possível a disponibilização de algum tempo para a formação dos envolvidos? Não ☐
antes do inicio do projecto ☐
durante o decorrer do projecto ☐
o projecto é delegado a outro elemento ☐
tempo médio para formação : _____

b. Já aconteceu rejeitarem algum pedido com base no seu grau de complexidade?
não ☐
sim ☐ logo ao início ☐ durante o desenvolvimento ☐
com base em: _____
_____
_____

c. Comentários:
_____
_____
_____
_____

5. No contexto do **desenvolvimento para a Web**
   a. Quais os pedidos mais recorrentes? _____
   b. Quais os serviços envolvidos? _____
   c. Quais as tecnologias envolvidas? _____
   d. A equipa média é constituída por _____ elementos
   e. Qual o cliente-tipo da dreamlab? _____
   f. Comentários:

6. Com a chegada de **novos pedidos**
   a. São começados de imediato ☐ ou ficam em lista de espera ☐ ?
   b. Equipas já formadas/a trabalhar podem ser fracturadas?
      não ☐
      sim ☐ com base em: _____
   c. Se o produto tiver serviços já desenvolvidos em pedidos anteriores
      projecto feito de raiz ☐
      reaproveitamento de funcionalidades ☐ como é o caso de: _____
   d. Comentários:

7. **Tecnicamente**, no desenvolvimento dos produtos
   a. Há uma preferência pelo uso de computadores
      pessoais ☐
      da empresa ☐ sempre o mesmo ☐ há rotação de postos ☐
   b. Há uma preferência pelo uso de linguagens de programação específicas?
      não ☐
      sim ☐ como é o caso de: _____
      _____
   c. Há uma preferência por software
      open source ☐
      licenciado ☐
   d. Qual o software usado mais regularmente? _____
      _____
   e. Comentários:
      _____
      _____
      _____
      _____

b. Entrevista ao gestor de projectos

[a entrevista anterior foi efectuada a uma programadora, esta será com a orientadora de projectos, para uma visão mais alargada e generalista acerca dos projectos. esta primeira parte serve ainda como complemento ao acompanhamento presencial, para melhor compreender o desenvolvimento dos projectos actuais no âmbito do desenvolvimento para a Web.]

I.

[na parte do desenvolvimento para a Web...]

a. Que projectos estão a decorrer neste momento?

b. Há já quanto tempo?

c. Quais as equipas por projecto?

d. Adicional:

II.

[ao dirigir as seguintes questões à orientadora de projectos espero conseguir respostas mais elaboradas e com mais conhecimento de causa. além disso espero também conseguir perceber a preocupação das chefias na questão do reaproveitamento de conteúdos]

a. Quando são encomendados os sites, quais os serviços mais recorrentes?

b. Na entrevista anterior passou a ideia de que a equipa de programação tenta sempre reaproveitar trabalho de uns projectos para os outros. Esta prática fica a cargo da equipa de programação ou funciona como uma directiva da administração?

c. Adicional: